2013

Improving Retrieval of Information from the Internet

Ruoxuan Zhao

Improving Retrieval of Information from the Internet

By

Ruoxuan Zhao

A Thesis
Submitted to the Faculty of Graduate Studies
through **Computer Science**
in Partial Fulfillment of the Requirements for
the Degree of **Master of Computer Science**
at the University of Windsor

Windsor, Ontario, Canada

2012

Improving Retrieval of Information from the Internet

by

Ruoxuan Zhao

APPROVED BY:

Dr. Jianguo Lu, Department of Computer Science

Dr. Rupp Carriveau, Department of Civil and Environmental Engineering

Dr. Joan Morrissey, Advisor

Dr. Alioune Ngom, Chair of Defense

18/01/2013

DECLARATION OF ORIGINALITY

ABSTRACT

To improve the quality of the search result returned by the internet which makes users have to look through a huge amount of links for the real answers, we utilized the high quality links Google produces and the Information Retrieval technology to implement a Question Answering (QA) system. This system analyzes and downloads the text contents from the relevant web pages Google searches based on the users' questions to build a dynamic knowledge collection; retrieves the relevant passages from the collection and sends the ranked passages back. The users can further refine their questions in the query refinement step for the better answers. A novel search strategy was designed to detect the semantic connections between the question and the documents. This answer retrieval also involves the TF-IDF algorithm and Vector Space Model for the document indexing. We have modified the original Cosine Coefficient Similarity Measurement to rank the candidate answers.

DEDICATION

To my parents who always fully support my decisions and taught me how to be an optimistic, responsible and kind person.

ACKNOWLEDGEMENTS

Great thanks to my supervisor Dr. Joan Morrissey. Thanks for your support, suggestions and most of all, the encouragements you have offered me for the past two years.

Thanks to my thesis internal and external readers: Dr. Jianguo Lu and Dr. Rupp Carriveau. Much appreciate your suggestions and recommendations about my research.

Thanks to people who kindly helped me during my study. Your kindnesses will never be forgotten by me.

TABLE OF CONTENTS

LIST OF APPENDICES

CHAPTER 1

A Survey on Question Answering System

Introduction

As the internet is no longer an information search tool just for the researchers to search for the academic information, more and more casual users prefer to use web search engines to find the information they need in their daily life. Moreover, due to the fully accelerated technology development, both the professional researchers and the casual computer users are seeking for the answers which have the higher qualities but cost less efforts. However, the ordinary and traditional online web search engines are only able to offer the relevant and ranked webpage links as their search results to the users. The users then have to spend the extra time on browsing the related web pages these links lead to until they find the expected information. For example, if a user types the question "who is the current president of Canada" into Google, Google will give back about 318,000,000 links in 0.32 seconds. This user is then overwhelmed by the huge number of links. The user will also be more confused because the top five pages do not offer the correct answer but first educate the user with the fact that there is no President but a Prime Minister in Canada. Base on this constantly happened example, a better answer search engine or system is highly needed to improve the users' information searching experience.

A Question Answering (QA) system is designed based on this motivation. Instead of feeding back the relevant web page links or the searched complete documents which require a deeper answer search to the users, an ideal QA system is able to answer the questions with the direct and correct answers. For the example mentioned above, the ideal QA system is expected to give back the answer which is similar to "Canada does not have a President. The current leader of Canada, who is called the Prime Minister, is Stephen Harper”. The users then will be corrected and answered at the same time by this answer.

There are basically three main components a QA system is usually comprised with. The first one is called question processing model. It is supposed to help the QA systems to “understand” the users’ questions. The second model is an answer matching model. The QA systems need it to retrieve the potential candidate answers according to the processed search queries produced by the first model. The last model is an answer validation model. This model should validate and rank those retrieved candidate answers and send parts of them back to the users in a certain order as the quality answers which will likely answer their questions. Those three models cooperate together by exchanging their internal process results to support a functioning QA system.

This survey collects, organizes and summarizes the newest technologies and approaches from 60 published papers in the QA area in the last five year (2008-2012). The section A focuses about the new methodologies and technologies which improve the performances of the QA systems’ different components. The section B is introducing the different QA systems in the different fields with a more general view of each system’s overall performance.

Section A. Methodologies and Technologies Implemented in QA Systems

In this section, some of the updated methodologies and technologies which are introduced in the papers published in the last five years are summarized. Each of them is presented with the structure as the motivation, explanation and evaluation. Also, the section A is distributed into three parts based on the three main tasks of QA systems: the question analysis, answer matching and answer validation. However, there are not perfectly clear boundaries among them because those three main system's components which are sending and receiving some intermediate information with each other are highly integrated. Some algorithms and strategies the writers came up with serve more than one components. But we still tried to classify them as much as possible in this survey to present a clear and organized structure to our readers.

1. Question Analysis

For the QA systems, the very first task they should accomplish with is to understand what their users are asking about. The traditional way the systems offer to the users to express their questions is giving them some limited choices of the question expressions. Some of the systems allow the users typing terms with the logic operators: "AND", "OR" and "NOT". This kind of questions then will be analyzed by the logical algorithms to form the binary search queries for the QA systems to retrieve the answers. The researchers describe this procedural as a question processing. They claimed that unless the QA systems can successfully deal with the users' questions in natural languages, they are still just processing them rather than understanding them. That is why Natural Language Processing (NLP) is the most popular technology in not only the QA systems but also other areas like the mobile phones, the artificial intelligence, etc which need to interact with human beings.

The algorithms and strategies are introduced in this part in the order of first some newly updated approaches of the syntactic analysis and then the higher level theories such as the semantic analysis and other new strategies in the NLP area.

1.1 Noise Reduction and Keyword Searching

A QA system named Short Messaging Service (SMS) is introduced by the authors in paper [1].The users can ask their questions by using their cell phone to send a short text message to the SMS server. This server then automatically analyzes the users' questions and extracts the proper answers to send back to the users' cell phone.

Two steps of the query analysis are emphasized in this paper. One is based on reducing the "noise" existing in the users' questions since the users type the message in natural languages. Once the server receives a query, it distributes each sentence into the different query tokens. Thus, the useless terms which are the noises in the sentences will be detected and eliminated. Furthermore, an algorithm called Longest Common Subsequence is used to make sure the noise reduction process is qualified and efficient while it is converting an original sentence into a noise-free search query. In other words, no valuable information should be missing after the noise reduction.

Also, the traditional keyword matching is involved into the answer locating of this system. The SMS uses a link parser to extract the key words from the processed queries. Those key words are going to help the system find the relevant answers in the correct category from the document collection which has been properly classified already.

1.2 WHY Questions

Dealing the different types of questions with the appropriate strategies is a big challenge in the question analysis area. Among the different types of questions, the WHY questions are especially unique: they usually do not contain a formal definition or tractable keywords such as "name"," date", "located" which can be found in the WHAT, WHEN and WHERE questions. Secondly, the answers to the WHY questions are usually distributed wider in the relevant documents than the HOW answers are. It is because the expected HOW questions' answers are the instructions in the related documents. They are better organized and easier to be located than the WHY answers which are the explanations. Therefore, searching for the WHY answers requires the QA systems to be implemented with some more sensitive and critical algorithms.

Unfortunately, most of the current QA systems do not perform well on the WHY questions based on the evaluations. Only the parts of the WHY answers can be recognized by the helps of some typical and mark-able key phrases such as: "the reason of", "therefore", "because of" in the document. The more causal and more implicit answers are unfortunately missed by the traditional answer retrieval strategies. Here, the traditional way to search for the WHY answers is manually pre-installing some typical question patterns to be prepared for the future questions' processing. The relevant information from the database then is extracted and collected to form the final answers based on the matched and filled question pattern. However, the hand-craft question patterns do not cover all the questions the users are going to ask. Moreover, the ideal WHY answers the system need to produce should contain the expected causes required in the users' questions. These are the challenges in WHY question answering.

In paper [2], the authors present a corpus-based approach to improve the quality of the WHY answers. The more general and causal expression collection of the WHY questions were built based on an existing corpora. Each piece of expression was also marked for a better recognition. That was how these causal expressions of the WHY questions are automatically learnt and prepared by the system.

Moreover, the researchers were using the existing thesauri to measure the similarities between the question and the answers. Thus, the useful answers which contain some untypical keywords which are similar to the tracking terms are not missed anymore. The researchers also pre-set the pairs of standard expressions stores the causes and the corresponding effects which are the core elements in the WHY answers. If a pair of a query and a candidate answer does not have a high similarity measurement value but they are detected that they share the same pair of cause-effect expression, they are considered as holding a causal and valuable WHY question-answer relation between each other. And this answer definitely needs to be retrieved back to this question.

Another approach the writers made was creating an answer ranking scheme for the system. This ranking scheme had been trained to be able to recognize the potential WHY answers. The training process starts with asking the scheme a testing question. The IR engine then is executed to retrieve a few relevant documents back based on the received question. The last step of this training is manually selecting all the correct answers among the retrieved documents. The selected answers then will be stored with the corresponding questions as the answering experiences which can be used in the real question answering.

1.3 List Question

Among the various types of questions the QA systems deal with, another unusual question type is the list question. The list questions request the QA systems feeding back a list of different and valid components to perform a complete answer. For example, the question “who were the prime ministers of Canada” apparently needs a list prime ministers' names as an ideal answer. Therefore, we define this kind of questions as the list questions.

However, the list questions are usually not answered completely and efficiently in the QA system due the lack of the NLP ability. The users have to find the complete answer by manually collecting some valuable segments from not only the first retrieved candidate answers but also the multiple searching results if it is necessary. Base on this situation, a number of approaches including a website called Google Sets have been made by the researchers for intending to solve this problem. One of them is called Set Expander for Any Language (SEAL). SEAL is an expansion of the QA systems which works after the systems searches back the candidate answers. SEAL helps the systems produce a more complete answer for the users by further analyzing the original candidate answers. During this analysis, SEAL acquires more keywords from the answers and executes another answer search base on those new key words in order to search more information to complete the original answer.

In paper [3], writers present some approaches based on SEAL. They have built up a component called Aggressive Fetcher (AF). AF receives the new key words talked above from the QA system's first searching result and sets each two of them as a team to form some new search queries. Thus, if SEAL extracts n number of new key words from the original candidate answers, AF then produces n*(n-1)/2 new search queries to make sure each keyword will be retrieved equally for building a more complete answer.

The researchers also came up a lenient extractor for SEAL. A lenient extractor only asks the system to search at least one new query at a time from the new query collection AF produced. It avoids the situation that there are not enough sentences are retrieved leaded by searching the new formed queries together. It makes sure this advanced answer search retrieves more useful information that the system did not acquire at the first time.

At last, the researchers found the key words which can be extracted from the original questions are also able to raise the quality of the SEAL search results. Thus, they created the Hinted Expansion (HE) to send the original key words from the users' questions to SEAL. Those original key words will work as the hints with the new search queries produced by SEAL for a better context acquiring result.

In their evaluation of using the SEAL expansion to answer a number of list questions from TREC 13, 14, 15, the targeted QA system' list answers were 55% improved. According to this, they claimed that their approaches were useful for improving the quality of the list answers for the QA systems.

1.4 Question Classification

To correctly recognize the questions' types, one of the most important components of the QA systems is the Question Classifier which conducts the task to classify the various questions into the different categories. It helps the QA systems produce the related answers by narrowing down the possible knowledge areas those candidate answers should be extracted from. However, it is also a big risk for the QA systems to take while classifying the questions. Once the received question is classified into a wrong category, it can be sure that the coming extracted answers will be totally irrelevant. Therefore, building up the explicit knowledge categories for successfully recognizing and classifying the different questions is crucial for the QA systems before mapping the correct answers to the questions.

In paper [4], the writers come up a “stand-alone rule-based” question classifier which detects the headword of a question and maps it into the corresponding knowledge category by using the WordNet. The WordNet stores abundant English terms and groups the terms which have the similar meaning. The detected headwords are not only those typical words such as “who”, “where” and “when” but also some identical terms which will help the systems have a clearer idea of which topics the received question describes about. Two main components were designed and implemented in this question classifier.

A question pattern matcher is used to locate some specific questions. For example: the question “who is Benjamin Franklin” is mapped into the category of Human: Description. A rule-based question parser is another component to identify the headwords of the questions and maps them into the question classification based on the WordNet. For example: “Which counties produce Avocado” The word “Avocado” is detected and the system is then going to search the relevant information from the category of fruit.

The researchers in this paper also use 60 rules to accomplish this question classifier. Those 60 rules help the classifier assign different questions into the correct categories. During the evaluation, the researchers found this question classifier provided an 87% precision by using the breadth-search strategy. Since the headword location was integrated with the pattern matcher and the question parser with the 60 advanced rules, it therefore performed better than the simple WordNet word mapping which offered 86.4% accuracy and 78.5% accuracy respectively.

1.5 Semantic Analysis

As the development of the question analysis technology in the QA system, the syntactic analysis is not helpful enough anymore for producing the accurate answers. It is because the surface pattern which is another name of the syntactic pattern only connects the key words in the question and the answers. For example: the surface patterns such as "<NAME> invented <INVENTION> on <DATE>" is a query which needs to be answered by the sentence in a pattern structure of "<SCIENTIST> <INVENTION> <DATE>".

Although it has been approved that the surface patterns are effective and accurate, they still have a few of disadvantages. First of all, only the limited numbers of the question patterns have been stored into the QA systems. More various and specific patterns are needed to deal with the unpredicted human questions, even though this also leads to the result of increasing the processing time of the question analysis. For example, <PHYSICAL INVENTIONS> offers more information than the simple pattern <INVENTIONS>.

Secondly, a syntactic pattern can only match to one of the topics of a received question due to its designed structure. For example, the question "who published the paper with the scientist invented the telephone" can only be matched with either the pattern <SCIENTIST> <PUBLICATIONS> or the pattern <SCIENTIST> <INVENTIONS>. Thus, the traditional pattern matching method cannot deal with the high distribution of the valid information in the question or the relevant documents. Moreover, if there are multiple languages involved in the questions, the QA system is only able to deal with the one which is used in its question patterns.

Therefore, in paper [5] and [6], the writers introduce their approaches which are aimed to solve some of these problems above. They imported the semantic information into the syntactic patterns in order to perform the semantic analysis to increase the accuracy of the questions analysis and the answer extraction. Thus, the labels in the question patterns also carry the semantic information with the machine-understandable format. Rather than concentrating on identifying the question types without using the question classifications, the researchers are focusing on generating the relations between the different semantic patterns to represent more questions' meanings. They also came up the rule that "the instantiation level of a semantic label determines the semantic capacity of the patterns". In other words, more specific a semantic label is, the more semantic information it holds.

The syntactic patterns which contain the semantic labels must also be supported by the hierarchy concept structures. The basic hierarchy structures mean a group of similar words is represented together by a father (super) term. Thus, the semantic labels can match more relevant terms and information eventually based on their similar labels acquired from the hierarchy structure. Those supported patterns help the answer extraction by mapping the similar semantic content from the documents base on the semantic labels. The researchers in the paper [6] claim that the semantic patterns work in both the automatic and interactive QA systems. And they have built up a user interactive QA system to implement their approach as well.

Furthermore, the implemented user interactive QA system assigns the semantic information to the syntactic patterns manually by the users. In order to ask the qualified questions in this system, the users need to firstly choose the questions patterns offered by the system which usually starts with choosing the question types. The users can even create their own question patterns by following the instruction in case of they cannot find the ideal patterns to express their questions.

Furthermore, based on the patterns the users chose, they will be asked to fill the blanks of the chosen patterns' labels with the semantic information. After they create and submit their questions, the system locates the relevant answers according to those manually created semantic patterns and presents them back to users.

Another similar question analysis strategy called question reformulation is represented in paper [7]. It is the core strategy of a reformulation-based QA system. It is about using the semantic constrains to increase the efficiency and accuracy of the answer searching.

The reformulation-based QA system converts the questions into the system-understandable queries. That is reason it is also called as a surface-pattern method. The semantic-based reformulation algorithm is about pre-processing the knowledge documents which were downloaded from the web pages and converting them into the pattern-based formats attached with some certain questions types. For example, the sentence "Ontario is the largest province population wise in Canada, and as of July 2009, the population of the whole province was 13,250,000 folks living there." is processed by the system and a new pattern will be produced as <PROVINCE> <POPULATION> attached with the question type "How many". Those question types are assigned by a training corpus which contains the pairs of the typical questions and answers. Those pre-processed knowledge is then stored in the database and waiting for the future matching to the new questions. If a question “how many people live in Ontario” is submitted by a user, this question will be analyzed and the key words “how many” will be detected to confirm the question type it is. The key words “people” and “Ontario” will also be matched to the knowledge pattern <PROVINCE> <POPULATION>. Thus, the processed knowledge mentioned above will be retrieved as a candidate answer to this question since they share the same pattern.

Another problem is, due to the term weighting strategy and other basic information retrieval techniques used in the QA systems, the text terms in the documents and questions are processed and distributed separately. Thus, the question “Who is the first person invented cars?” never gets the answers which contain the word “vehicle”. This situation has been found as quite normal in the existing QA systems.

In paper [8], the researchers come up with a new strategy by using the Language Modeling (LM) technique to improve the situation talked above. The LM technique aims at analyzing the syntactic and semantic meaning of each term and clustering them to form a better retrieval result. The original approach had been made for years by other scientists. But the researchers in this paper claim that it was difficult to estimate the relations between the different terms and use those relations into the term weighting scheme.

Therefore, they made some experiments to compare their new model to the original model and the result showed their model could help the sentence retrieval get a precision of 23.62% to 29.91% which was better than the original one did.

In paper [9], a new Question Answering System which is named as Question Answering for Automatic Learning (QAAL) system is developed and introduced by the writers. This QA system is ontology-based system which means the information in the database is pre-processed and stored with a highly structured format. It provides the answers to the users from the semantic view.

Since the ontology methodology is used in the closed-domain QA systems in most of the cases, the writers in this paper believe the knowledge representation and the logic inference techniques are crucial to the efficiency of their QA system. Three main steps are explained in this paper: question analyzing, information retrieving and answer extracting. Among those steps, the question formalization is emphasized that this technique is based on a conversion mechanism for the usage of the ontology methodology.

The first step of formalizing the questions written in a natural language is to collect some existing questions in different knowledge areas. These collected natural language questions then will be labeled with the annotations indicating the different categories they belong to. Thus, the QA system is able to create the different question patterns according to the collected and processed questions. The researchers made this achievement with the help of the Word Segmentation Process. This process can be also seen as the knowledge training process. Moreover, the QAAL system uses the Resource Description Framework

(RDF) data structure which helps the system work with the ontology methodology. Since the SPARQL is one of the RDF query languages, those question patterns the system created need to be transformed into the SPARQL form for the future question matching.

During the evaluation of this QAAL system, the writers found this system worked well only with the predicated questions. This was not a surprising result since the question patterns are created based on the collected and prepared questions. The various natural language questions certainly need more flexible patterns to be processed with. As the result, the researchers claimed this was the disadvantage of their QA system which needed to be improved in the future.

In paper [10], the writers are also focusing on utilizing the category information to match the queries to the relevant answers. For accomplishing that, several algorithms are involved to convert the questions which are written in the natural language format into some formal search queries which work well with a traditional IR model. The basic process steps of their QA system are introduced below:

1. Receive a question written in a natural language from a user.

2. Analyze all the sentences in this question and convert them into the predicate logic sentences.

3. Convert those predicate logic sentences into a clause form.

4. Base on the results generated by the step 2 and 3, acquire the useful semantic information from them.

5. Use the resolution and unification algorithm to retrieve the relevant documents stored in the database base on the extracted semantic information from the question.

6. Measure the similarity between the query and those retrieved documents to evaluate the best answer.

7. Feed those qualified candidates answers back to users.

There are some important details need to be mentioned among those steps. For the step 2, the predicate logic converting algorithm identifies the subjects by removing all the unnecessary and disturbing terms such as “and”, “for” and “in” from the question sentences. The clause forming in the step 3 is aimed at converting the targeted question into a logic based chain structure for the further document retrieval.

1.6 Question Templates

Unfortunately, the technique which is dealing with the natural language questions the users type into the QA system has not been fully developed yet. There are still many misunderstandings about the users’ requests. The users’ satisfaction about the answers given back from the QA systems is still very low. While other researchers are keeping developing some better approaches of the NLP technology, the writers in paper [11] are trying to solve this problem by a novel strategy. Instead of using the syntactic and semantic analysis to understand the questions, they came up with a question template which was used to match the new questions.

The question template was created based on a few close-domain QA systems. Those systems store the pairs of questions and the corresponding correct answers in some specific areas based on their question answering experiences. The researchers then collected and imported them into their own open-domain QA system and pre-organized them into the relevant question categories. Once there is a new question asked by a user, the system automatically maps this question to some of those stored question templates and calculates the similarities between the received question and the mapped similar temples. The question template with the highest similarity score then will be picked up and its corresponding correct answer will be given back to the user as the answer to the question.

The writers also installed this application into the mobile devices to test it in some specific consulting areas. The result was very outstanding: this QA system could “understand” 97.8% of questions it dealt with and the answers it gave back offered the accuracy of 82.4%.

As a matter of fact, these figures were quite predictable since the prepared questions were stored with their correct answers in the system earlier before the test. In most of the cases, the correct answers were given back only when their corresponding question templates matched to the test questions. In other words, it made sure the system’s answers were perfectly correct as long as it marched the truly relevant question templates to the test questions. That is why the accuracy was higher than usual.

However, with a long-term vision to consider this case, it actually did not solve the basic problem of the QA systems which is dealing with the variant and random questions written in natural languages. It still prepares a limited and static knowledge source for answering the users’ unlimited questions. Eventually, this system will be either having an unacceptable long processing time with a huge question-answer collection or confused by more and more questions they have not been prepared before.

2. Answer Matching

In this part, the techniques about matching the relevant documents from the local document collections or online web pages and extracting the more specific candidate answers from those searched documents are represented. As a matter of fact, the answer matching techniques are not working as an independent component in the QA systems. Other components such as the query formalization which is also called the question processing and the answer ranking also cooperate with the answering matching component deeply. In other words, the answer matching model can receive the relevant information and produce more accurate answers if and only if other related QA systems' components accomplish their tasks. Therefore, in this part of the answering matching, there are still some other components involved and mentioned to offer a clearer view of the answer matching process.

2.1 Ontology Methodology

For a better understanding of the natural language written questions and a higher quality of the candidate answers, the ontology methodology is used widely in the different QA systems as the core strategy for analyzing the questions and searching the correct answers. There are basically two types of the ontology: domain ontology and upper ontology. The domain ontology which is also named as domain-specific ontology only offers a specific domain which contains limited topics or classes and the relations among those classes to represent the stored knowledge. For example: the scientific publication domain may have several classes such as the name of the paper, the scientist's information, the institutions' names, etc. Those different classes work together to store the relevant information to present a complete scientific publication information database. The domain ontology is usually used as storing the annotations of the terms in a specific field and then analyzing the natural language written questions based on those annotations. The Upper ontology such as WordNet and OpenCyc focuses on some concepts in the unlimited areas. Thus, the upper ontology covers more topics than the domain ontology does.

Overall, the objectives of using the ontology in the QA systems are creating a practical dictionary for each certain domain and mapping the questions to the relevant answers together based on that dictionary.

In paper [12], the writers introduce a special ontology called QALL-ME ontology which is about the tourism domain. Specifically, it focuses on the static tourism information like the hotel reservation and the event promotion rather than the dynamic information such as the trip scheduling. Moreover, QALL-ME is claimed has a larger coverage which means it contains more tourism websites' and festival events' information than others systems do. Since it deals with the online information of the hotels and flights websites, it was designed with the Web Ontology Language (OWL) which is the most recently updated ontology language. Therefore, more classes and relations are involved to represent the valuable information and they make this QA system more complicated.

Furthermore, the QALL-ME ontology is also mapped to some upper ontology to exchange the information for a broader coverage. The traditional similarity measurements are used to locate the relevant words between those two types of ontology to accomplish this mapping process.

Overall, the QA systems are able to easily match the analyzed questions and answers together according to the definition of the ontology: it formally creates the conceptual representations of some relevant knowledge and their relations in a specific domain. However, the biggest problem which has been slowing down the development of the QA systems is the various natural languages the users use to express their demands. Understanding and translating those questions are very difficult and complicated. In paper [13], the writers present some approaches to reduce the gap between the questions and the knowledge.

User Query Formulation (UQF) database is one of their main approaches. The researchers analyzed some existing questions by collecting and clustering them together into the same domains before receiving the users' questions. Having those clustered questions in the database helps to deal with the new coming questions by matching them with the stored questions. During the question matching, if there are two questions sharing the same topic but they are written in different expressions, the system is able to detect the common topic and use this captured domain information in the coming answer extraction. The last step of the matching process is a semantic deduction between the new received question and the similar one stored in the database.

After an ontology-based QA system finishes analyzing the natural language questions, there will be more than one model in the system working on the next answer mapping process. Especially in the open-domain QA systems, more models are applied to be used for attracting the better answers from the large knowledge domains. In most of cases, the multiple-model processing strategy is used to execute those models separately and receive back the different analyzed results. The QA systems then combine those internal results carefully to form the retrieved candidate answers.

In paper [14], the writers claim that this multiple-model processing strategy sometimes is not efficient since some straight forward questions such as “who is Barack Obama” do not need a complete and sophisticated answer retrieval process to get the correct answer. Furthermore, for some QA systems which are manually applied with some models still require the developers to have a deep understanding of the targeted knowledge area to be able to tune these models later. Therefore, those QA systems should only be the closed-domain systems to avoid of dealing with the overwhelming great amount of information with several sophisticated processing models. Also, manually applied models require another huge amount of efforts for the updating or maintaining. Therefore the writers in paper [14] present some approaches to invoke each model in a QA system separately and conditionally. They found their strategy which routes queries with the chosen models saved their QA system 27.2% efficiency and improved 10.5% effectiveness.

The Business Intelligence (BI) applications allow users using their queries to acquire some valuable information to help themselves make their business decisions. All the information is analyzed and stored in a Data Warehouse (DW). The DW combines the information from the different databases and the business data with the various formats. Nowadays, a new type of data which is described as the unstructured data is becoming the main data we process with. It is a new challenge because the DW used to utilize the traditional IR technique to only deal with the structured data. Since the IR techniques only search back the related documents, using the m to process the unstructured data will lead the result that the users will have to spend more time on reading the retrieved documents which contain the unstructured data and finding the expected answers. Clearly, the unstructured data requires the system be updated some new IR strategies.

In paper [15], the researchers integrated their QA system with the DW to produce a better search result on the unstructured data. To accomplish their goal, they chose the ontology methodology. Not like other approaches which use the ontology only for understanding the users' questions and extracting the relevant information from the database, the researchers used the information from both the DW and the received questions to label some extra concepts to serve for the detected domain searching. For example: the users use a BI application to search for the lowest price of a flight with a certain departure date. The QA system will search the weather temperature on that specific day and the BI will analyze this weather information with the regular prices and the sales in different companies from different database to predict the lowest price the users can get. Thus, the weather information, the deals of the flights is the extra information serving for the prediction of the future fight.

2.2 User Interactive Answer Matching

One of the development branches of the NLP technology in the QA area is the Interactive Question Answering (IQA) development. The IQA conducts some text dialogues to enable the users getting more involved with the query forming and the answer mapping processes. The users can set more search constrains to their queries by answering some instructive questions asked by the QA systems. Thus, the dialogue designing is very crucial to the IQA systems. Nowadays, the IQA technique is more likely being used in some online self-service websites such as the flights booking and restaurants searching websites.

One of the most important issues the IQA technique faces is how many questions a QA system should ask to its users to get enough search constraints for constructing the valid and robust queries. Clearly, the number of questions which are asked during the users' searching experience affects the answer search efficiency and the quality of candidate answers [16].

This considered fact is also affected by the different devices which are integrated with the IQA technique. As the mobile devices are becoming more and more popular and smaller, embedding the IQA system into the different mobile devices depends on different situations. For the GPS service in the vehicles, the writers of paper [16] noticed that it always involved a lot of the users' input noises and this service should not over disturb the drivers while they are driving. Therefore, the IQA on the GPS should be simplified on its interaction step. It should also be able to eliminate the unintentional input noises from the users. On another hand, the personal computers or other devices at home are assumed with the larger screens and the better user interfaces. The IQA technique in those devices can be optimized and allowed to have more dialogue models to deeper process the users' queries.

However, one of the problems the different IQA techniques are having in common is that their users sometimes are confused by the query refinement step. They are overwhelmed that they have to consider both the refinement decisions and the ultimate results to make the appropriate revisions. This situation usually leads to a result that the users set too many constraints that the searching results are either too small or not even existing.

In paper [17], the writers tend to perform a relaxation process on the searching queries which has been attached with the customized constrains by utilizing a content optimizer. The content optimizer accomplishes the relaxation task by modifying the values of the constraints. Sometimes it even just eliminates one of constrains in order to help the system retrieve more candidate answers to the users. As a matter of fact, this constrain relaxation is made based on the ontological relationships. For example, relaxing "the brand of Benz" restraint on a query means to delete it or replace it with its super class such as "German cars" which is stored in the vehicle-brand concept of the domain ontology.

Moreover, a linear constraint can be relaxed by adjusting its variables' values. Specifically, the content optimizer relaxes a binary constraint by reassigning it with a reverse value. For example, the binary search query "cook AND pizza NOT Italian" may not retrieve enough information about how to cook a pizza. It is because the most of pizza recipes are involved with the term "Italian". Therefore, the content optimizer can relax this binary constrain by replacing the operator "NOT" with the "AND" in order to help the query retrieve more pizza recipes.

The another job of the content optimizer is counting the amount of retrieved candidate answers and producing a proper threshold to only show a reasonable number of results to the users to view. For example, if a user searched for a Spanish restaurant for a dinner, the system will only show the top 10 restaurants which are qualified to user's constraints instead of feeding back all of the qualified restaurants' information.

2.3 Conceptual Graph Formalism

In paper [18], the writers present their QA system which is the implementation of a novel strategy called Conceptual Graph Formalism (CGF) to match the relevant answers to the users' "What", "When", "Where", "Who" and "How" questions. Similar with other QA systems which were designed in the last 5 years, this QA system is also accomplished with the syntactic and semantic analysis based on the VerbNet and the WordNet.

In the CGF theory, the Authors believe that each pair of question and the correct answer has its own conceptual graph representation. Those conceptual graphs are generated by some carefully designed formulas and will be used into the question answer matching process by a projection operator.

For the details, this QA system firstly splits the potential documents into the sentences and uses the NLP technique to parse those sentences in to several tokens. For each sentence, the system maps its each token to the different concepts categories using the WordNet ontology. Then, it creates the conceptual relations between each mapped token according to their syntactical relations to get a conceptual graph for the whole sentence. Furthermore, this QA system uses the sentence connector to also build up the conceptual relations between the different sentences and produce the ultimate conceptual graph of a whole document eventually. This ultimate conceptual graph is the result of the document analysis. It represents the information contained by each document and will help the projection operator match the users' questions to the related document.

The result the researchers received from their experiments in this paper emphasizes the importance of the syntactic and semantic relations between each sentence for the answer retrieval. They have been proved that their approach works better than the key word matching. They have helped the QA system understand and map the better answers to the questions.

2.4 Reasoning

In paper [19], the QA system is built base on a strategy called Case-Based Reasoning (CBR). The CBR is about answering the users' questions by searching the similar questions which have already been solved and stored with the corresponding answers in the database before. Thus, the CBR technique requires a large scale of the historical case database. Also, the techniques such as the automatic segmentation, the question similarity calculation are still the main procedurals in this system. The writers claim in the paper that their system has higher accuracy and efficiency based on their experiments.

As introduced above, the CBR as the core technology mainly depends on the experiences of the QA systems. Therefore, each solved case is stored into the corresponding categories based on the domain analysis. The basic procedurals of this QA system are shown below:

a. Input the query.

b. Analyze the queries: parse the questions, locate their key words.

c. Use the keyword search strategy to match some solved cases in the database to the received query.

d. Rank the candidate solved cases by calculating the similarities between the query and those cases.

f. Determine the value of the calculated similarity of each candidate case with a certain threshold indicating whether those existing cases are qualified to be shown back to the users.

g. If there are qualified cases, then show them to users. Otherwise, execute the full-text search module which performs a basic keyword search based on the original query. This backup plan makes sure the system always answers the users' questions.

2.5 Data Region Matching

For neither the open-domain nor the close-domain QA systems, building up an answer catalogue to store some existing correct answers is not easy. The way that distributes the different knowledge into the different categories affects the systems efficiency and answer accuracy directly. In other words, if the catalogue is not well designed or the distribution process is not perfectly correct, the QA systems will match some completely wrong answers to the questions. Thus, in paper [20], the authors present an open-domain QA system based on the catalogue of Wikipedia which is the largest online encyclopaedic question answering community. Furthermore, the writers also invented a dialog-based user interface as a virtual agent to interact with the users in Germany.

Their evaluations showed that their QA system provided 44% accuracy which proved that the catalogue created based on Wikipedia is useful and accurate. This catalogue improved the performing of this QA system.

Another data region marching QA system is the computer-based system in paper [21]. It is because this system is able to identify the regions of the data set based on the regions' annotations. Once the targeted data region is located in the database base on the users' queries, the system shows the details of this region to the users and also some potential answers from the same region. The users will also be asked to give the feedbacks to the system for helping it change or narrow down the searching regions.

2.6 Data Template Matching

Resource Description Framework (RDF) is a standard and popular model for the data interchange through the Web. The RDF merges the different data together without considering whether their underlying schemas are different or not. The merged data set is named as the RDF data. In paper [22], the writers come up a new way to retrieve the RDF data using data templates.

Comparing with the current QA Systems in the whole industry, most of them utilize the similarity measurements to process the users' questions and distribute those questions into the different question patterns. Then they match those patterns with the related documents stored in the database to retrieval the expected RDF data stored in those searched documents. However, due to complexity of natural languages, the question patterns in some cases do not work well to represent the meaning the users are trying to express. Some regular questions like “which city is the capital of Canada” works fine with the pattern representation. Other “non-traditional” questions like “which university has more than three presidents coming from” confuse the systems and they will probably be distributed into the wrong patterns. Clearly, the more restrictions involved into a question, a higher intelligent processing method is needed for the question analysis. Therefore, the authors were trying to solve this problem by designing a template called SPARQL template. They claimed this data template can directly indicate the internal structure of the users' natural language questions. In this paper, they claim that their approach decreases the users' answer searching time. The users do not have to pay the extra attentions on the grammars and vocabularies used in their questions. It has also been proved that this data template improves the quality of the retrieved answers.

2.7 Passage Retrieval

Passage retrieval is an important step of the QA systems. The systems use the input questions as the search queries to search the relevant passages in the stored documents and send the qualified passages back to the users as the candidate answers. However, in most of the cases, lots of the irrelevant paragraphs are also sent back to the users because they contain the common words which occur in the questions as well.

This result is the side effect of simply using the key word search strategy to retrieve the related answers. The QA systems retrieve the passages without checking the syntactic structure and the expected topic between the received question and the searched answers. For example, when users are asking the question "who is Jason Mraz", they expect some answers whose topic is an introduction of a person. Thus, the expected topic the QA system needs to use to determine the relevant passages is <person>.

In paper [23], the researchers implement this topic retrieval approach in their QA system. The questions' syntactic and semantic structures are analyzed in order to acquire the correct topics. Unfortunately, instead of getting an expected high precision from the searching results, they found their approach did not extraordinarily improve the search results. Due to the immaturity of the syntactic and semantic analysis, there were still some misunderstandings on the topic extractions. But the researchers still believe this approach theoretically works well. As long as they have a better strategy of the topic location, they will have a better answer retrieval result.

3. Answer Validation

The quality of the answers the QA systems produce is the most crucial factor which affects the users' answer searching experience. The question validation is the last procedural to evaluate the candidate answers which are about to send back to the users. Therefore, the question validation is getting more and more attentions in the QA system development. However, there is still not a mature and efficient strategy to check the retrieved answers especially in the open-domain QA systems. This is because an open-domain QA system usually has a larger knowledge database and deal with more information during the answer retrieval. Moreover, for the systems which classify the potential knowledge into the different categories, it is impossible to accomplish this knowledge classification perfectly. Thus, we need a more dynamic answer validation strategy to determine the real potential answers.

In paper [24], the researchers are trying to combine three different answer validation methods together to get the better search results. The first method is utilizing a static way to check the question type on those potential answers. Only the answers which are the same type with the question will be considered as qualified. For example: for the "WHO" questions, it is clear that only the answers which describe persons are the expected answers the systems should look for. And those answers should be the only qualified candidates pass the validation procedural. The second method is also a static method which compares the topics between the question and the retrieved answers. After successfully locating the main topic in the question, the method calculates the occurrences of the detected topic in each candidate answer. A higher occurrence of the topic in a detected answer indicates a better answer quality this answer serves. The last method relies on Wikipedia. Wikipedia offers a classic and sophisticated strategy to determine the answers type. Researchers used this feature to help their system identify the correct answers.

Furthermore, the researchers combined the evaluation values produced from those three methods to have a complete view about each candidate answer. They claimed that their strategy gave back a satisfied result. However, this approach only collaborates with other answer validation techniques. It is not able to locate a correct answer by itself.

In paper [25], the researchers compare two different classification methods of the answer validation. One is bases on a traditional way which is using the precision and the recall formulas together to formulate a new answer quality measurement called F-measure. Another one is using the analysis of Receiver Operation Characteristic (ROC) space summarized in Area under the Curve (AUC) scalar value measurement.

Both of those two measurements are based on the binary classification which means there is only one value indicating the status of a candidate answer: correct or incorrect. In this paper, the writers compare those two measurements and the result is quite successful. They found the F-measure worked more stable than the AUC measurement during the experiments. Also, the F-measure was more discriminative for detecting the correct answers. However, the authors indicated that those results did not prove that F-measure was a complete better choice. It still depends on the different QA systems and the users it is tested with. If the users are looking for a measurement with a stable processing ability which produces some obvious results, then they will feel more comfortable using the F-measure. However, there are still some extraordinary futures of the AUC measurement. It is less sensitive to the changes of the answer collection. Thus, the AUC measurement is more useful to acknowledge in the same way with both the correct and the incorrect answers.

Section B Presentations of QA Systems

In section B, more than 30 new QA systems are introduced. The papers presenting those QA systems are more concentrating on the overall performance of a complete QA system than the system core strategies and the new approaches. This is also the main difference between the section A and B. Here, the QA systems are summarized and organized as much as possible based on their different device environments, processing languages and various usages. The last part of this section is named as "other QA systems" which is representing some valuable and interesting ideas in the QA area. They are hard to be classified into any categories but still worthy to be viewed.

1. Biology QA System

In paper [26], the writers present a survey about the biomedical information QA systems. Due to the great developments and demands in the biomedical area, research in the corresponding domain QA systems has never been this popular. However, answering the correct answers to the questions is even more crucial than other QA systems since the questions seek the academic knowledge in the health care area.

Therefore, the researchers in this paper point out there are quite a few differences between the close-domain QA system and the open-domain QA system which affect the design of the biomedical QA systems. They found that the close-domain QA systems better deal with some specified knowledge areas and retrieve the deeper information for the knowledge resource than the open-domain QA systems do.

Another fact needs to be considered in the system designing is the special characteristic of the biomedical-domain QA systems. This kind of systems more likely receive some un-professional questions for their users than other systems do since the biomedical knowledge is more professional and difficult to understand. Thus, dealing with the bigger gap between the non-expert users' questions and the professional documents is the biggest challenge of developing such systems. Unfortunately, besides the ordinary techniques such as the NLP and the IR technologies, the important utilities of the logic representation and the inference mechanism have not attracted more attentions from the researchers in this area.

2. Monolingual and Multilingual QA Systems

The languages are the key points to a QA system. The questions asked by the users, the documents stored in the database and the retrieved answers sent back to the users are all written in one or more particular languages. Furthermore, the different branches of the languages have their own features that only the corresponding professional language users are able to process with. Therefore, the researchers from different countries are making their own efforts in their QA systems using their own languages. Some multilingual QA systems are also designed for sharing the knowledge written in the different languages due to the information globalization.

2.1 English Language QA System

In paper [27], the writers introduce a QA system developed by them which focuses on an approach of extracting the exact answers from some relevant documents to the questions. The language they were dealing with is natural English. The core technique which is used in their system is called pattern learning. More details of it will be explained below.

First of all, the input question is separated into several independent terms. Those terms are then stored into an array and will be processed base on their original order. By the help of this term order, the QA system is able to detect the different syntactic components in this question. For example: if the system detects a subject component in the question, then the next following component will probably be a verb or an adjective. In the meantime, all those high-frequency words such as “in”, “at”, “on”, etc. will be eliminated from the term array because they are useless for understanding this question.

Before the processed term array is converted into a clause in which the detected syntactic components will be connected with some logic operators, there is one more algorithm called Unification algorithm to further process this sentence. This algorithm finds the simplest substitution to convert two similar terms into one. This step simplifies the term array before the answer matching. Thus, this term array is carefully simplified and converted into a logic clause. The QA system uses this formal query to match the relevant information from the database separately to each component in the clause. Then those retrieved information segments will be combined together based on the question’s analyzed syntactic structure to form an exact and complete answer to the users.

2.2 Chinese Language QA System

In paper [28], a closed-domain Chinese cuisine QA system is introduced. It uses the question classification technique to narrow down the relevant candidate answers. It also determines the appropriate strategy to extract the answers from the documents. In this QA system, the researchers also designed a question catalogue based on the core information the users' questions carries. It is similar with the document classification technique. Without having the situations that the QA system is confused by the questions due to the lack of context information, this restricted-domain system has been successfully designed debased on the Chinese cuisine. The input question is first determined as which question type it is by the production rules. Then it is matched with the filtering rules according to its domain attribute. If the matching is not successful, a deeper secondary classification will be executed base on the machine learning method. The designers of this QA system found that building up the question taxonomy and setting up the question classification rules are the most challenging parts of their works due to the features involved in the cooking area.

2.3 Germany Language QA System

The authors in paper [29] design a QA system named LogAnswer. LogAnswer is an Internet-based German language system. This means LogAnswer acquires the necessary knowledge from the Internet. The writers claimed that they had successfully integrated the AI, NLP, machine learning, knowledge representation and automated theorem proving techniques in their system. Moreover, they came up a machine learning approach which deals with Wrong Answer Avoidance (WAA) problem. They have accomplished this approach by adding a term-based classifier and a dictionary which stores all WAA featured terms in their system. Once there is a wrong answer which is accidently searched based on a question, the classifier will be able to detect this answer and its identical terms by the help of the WAA dictionary and stop it from being presented back to the users.

The InSicht in paper [30] is a another German language question answering system which has been updated and integrated with several new techniques including a deep answer producer, a shallow question producer and a logical answer validator. It has also been implemented with a parser to analyze the natural langue questions. This parser firstly processes questions based on a semantic representation. Later on, it gives the inferences and attaches them on those representations. Those representations then are ready to be matched with the relevant answers. Furthermore, InSlicht is not only able to search for the knowledge written in Germany; it also has been proved that it works with English and Spanish documents in the database as well.

2.4 Arabic Language QA system

As one of the written languages in the knowledge resources and the search queries, Arabic language is not popular in neither the NLP technique nor the QA areas. However, in paper [31], the researchers are highly emphasizing the significant influence of the Arabic QA techniques' developments. They introduced a system called DefArabicQA. This system answers the definition questions which are also known as the "What" questions in Arabic.

Here are the basic steps of DefArabicQA: analyze the questions by using the question patterns; retrieve the relevant passages from the web resource; identify the candidate answers with some pre-learned answer patterns; rank those potential answers and feed them back to the users. Apparently, both the question and answer patterns are made based on Arabic. They contain some special features comparing with the English patterns to better detect Arabic.

2.5 Slovene Language QA System

In paper [32] a close-domain QA system with the knowledge written in Slovenian is presented by the writers who developed this system. Like other languages in the world, Slovenian language's features gave a hard time to the researchers to process the questions and retrieve the answers. Therefore, the writers claimed that their system is able to only partly solve some language difficulties of Slovenian. Moreover, this QA system was designed for the students to ask their academic schedule questions. It was aimed at releasing some pressures of the campus faculties.

Moreover, this QA system has a local knowledge database in which a small amount of knowledge has been pre-processed with the knowledge representation methodology. It also keeps an access to the internet to acquire the information from the semantic web pages. The writers mentioned that their system was capable for adding a dialog user interaction component for a better answer quality. Specially, they have set up two separate interfaces for both the users and the administrators. This allows the developers porting and training the new knowledge and the patterns without editing the code.

2.6 French Language QA system

Finding in Documents Justifications and Inference (FIDJI) is an open-domain QA system in French. The researchers who created FIDJI have integrated the syntactic analysis and the traditional QA techniques in this system. For the details, the named entity recognition and the term weighting strategy are the traditional technologies talked here. The writers in paper [33] introduce their approach which is aimed at accomplishing the same task FIDJI originally does but without accomplishing the heavy knowledge pre-processing task with a huge document collection.

In FIDJI, the question written in French is firstly analyzed and decomposed into the keywords, answer type and extended answer type by the syntactic analysis. The keywords then are used to locate 100 relevant documents in the database. Those 100 documents then are processed into the syntactic analysis and the named entity tagging until they are distributed as separate targeted sentences. Those sentences should contain the highest number of the common keywords with the questions. Then they are uniformed according to the syntactic relations and transformed into the candidate answers. Finally, an evaluation is executed according to the answer type acquired from the question at first step to determine if those candidate answers are valid or not. Only qualified answers are sent back to the users to be viewed.

2.7 Macedonian Language QA System

In paper [34], the researchers focus on a particular language: Macedonian in the QA systems. Since they believe that the different languages have their own features which should be fully aware in the system development, they have carefully created an appropriate test collection for evaluating the performance of the Macedonian QA systems.

They used their test collection to test the vector space model, a traditional model works for the similarity measurement, with the pivoted document length normalization. The results showed that this text collection is able to properly evaluate the performances of the Macedonian QA systems with the multiple test questions.

2.8 Romanian Language QA System

For Romanian language, there is a QA system called RACAI's QA system which is introduced and explained in paper [35]. It is a pattern-based system with a Boolean searching engine. The document collection is from Wikipedia in Romanian. Those documents have been pre-processed by decomposing them into the different tokens and tags. RACAI has two sub-systems. Sub-system A utilizes some traditional strategies to analyze the questions and convert them into the formal search queries. It distinguishes the noun phrases which create the content words and the noun phrases which follow after the content words. Different from the sub-system A, the sub-system B uses the Lex-Par [36] to acquire a dependency linkage which determines the topic of a question. This procedural is based on the grammatical patterns which form a dependency chain to represent the meaning of the processed question. The writers of this paper claim that RACAI with these two subsystems performs overall 30% accuracy and they find it was working well particularly for answering the definition questions.

2.9 Dutch Language QA System

A close-domain Dutch language QA system called Joost is introduced in paper [37]. Joost answers the questions in the topics of navigations and pictures. It deals with monolingual Dutch and multilingual English to Dutch questions. For analyzing the questions, it has been attached with a question parser called Alpino [38]. Alpino is designed for parsing the documents and sentences particularly in Dutch. The answers are then mapped by the question patterns produced by Alpino with the syntactic dependency relations. The syntactic similarity is also calculated between the question and the potential sentences to rank the candidate answers. Another special function Joost has is saving the correct answer and question pairs off-line for the future similar questions.

More specifically, the XML-version Wikipedia in Dutch is pre-processed and stored as the documents collection. During the document processing, the researchers only saved the documents which were about the navigations and the pictures into its database and distributed those documents into the title, main body and list for building up the information retrieval index.

For query formalization, Joost utilizes two strategies to deal with the natural language questions. It firstly converts the questions into the search queries based on their key words. Those keywords will also be weighted with a genetic algorithm. Secondly, there is a procedure called query expansion. It is executed for constructing a more robust query. During this step, two methods: global method and local method are used to find the relevant terms for expanding the question. In the last, at most 10 new keywords will be added on the original question to retrieve less but more correct answers.

2.10 Multilingual QA System

In paper [39], the writers test their Basque QA system Ihardetsi with three different languages: Basque, English and Spanish. They first used a machine translation system to translate the English and Spanish questions into Basque and then tried to answer them. For the question analysis, the writers utilized the question classification strategy to locate the keywords and the time information in the users' questions. As a cross-lingual system, Ihardesti did not perform well enough answering the test questions in different languages other than Basque. It is because some valuable information was lost during the machine translation step due to the different language structures.

In paper [40], an open-domain system called PowerAnswer QA system is integrated with a static machine translation engine for the multilingual question answering. As a qualified QA system which is capable for integrating and managing with different processing models, PowerAnswer is claimed as a fully-modular and distributed multi-strategy QA system. It has been attached with several powerful components such as: semantic relations, inference abilities and syntactically constrained lexical chains. In the meantime, the translation engine was defined as the open-source Phramer system [41]. It exploits a phrase-based machine translation algorithm for dealing the questions in different languages.

When PowerAnswer is dealing with the questions in English, French and Portuguese, it translates the input queries into English and matches the relevant English passages to this translated query. Then those English passages are retrieved back to the source documents and search the final answers for the users.

3. Medical QA System

In paper [42], the authors claim that dude to the requirement of the high quality medical topic information, the clinic-domain question answering systems need to be accomplished with the higher level techniques rather than the simple syntactic analysis. Thus, the semantic information should be imported carefully into those systems. Based on this realization, they presented a question classification strategy for the clinic domain QA systems.

Different from the ordinary question classification strategies, the researchers used a semantic categorization scheme to determine the questions' type again after the questions are simply classified based on their key words such as "HOW", "WHY", etc. They created four levels of the questions categories. The first level is based on the semantic relations detected in the questions. For example: the keyword "symptom" will lead the detected question to a specific category. The second level of the question classification focuses on the semantic classes' types. For example: the key word "transfusion". The third level depends on how specific the question is. It has only one logical factor with the value of yes or no as the indicator of the analysis result. The fourth level determines whether this question considers a particular situation. Thus, it is either general or contextual. This re-classification procedural performs a better questions analysis specifically on the close-domain knowledge such as the clinical questions in our case.

Another clinic question answering system is made and introduced in paper [43]. The writers named their system as AskHERMES. The main task of AskHERMES is semantically analyzing the complicated and professional clinical questions and giving back the correct and summarized answers.

A few professional document collections are involved: MEDLINE abstracts, PubMED Central full-text articles, eMedicine documents, clinical guidelines and Wikipedia articles. They are the knowledge resource of AskHERMES and providing the correct and professional information to the users.

As the evaluation of their clinic-domain QA system, the writers compared AskHERMES with the Google search engine and the UPTODate clinical database system based on the quality of the produced candidate answers, the system processing time and the users' using experience. The result of this comparison showed that AskHERMES offered an overall satisfied performance. It apparently had its advantages when dealing with the complicated clinical questions. AskHERMES gave the better answers to those questions.

Furthermore, in paper [44], the researchers presented a system which offers the decision supports for the doctors' medical diagnosis and treatments based on the medical knowledge. This system is mainly made of three models: an input model, an output model and a question answering model. The input is the description of patient's individual situation. This information will be converted into the formal search queries to seek for the possible medical diagnosis or treatments prepared in the database. The system then calculates the medical evidences detected in those queries and assigns a confidence value to each potential diagnosis or treatment by the help of the output model. Those candidate answers are then ranked based on their confidence values and sent back to the doctors as some appropriate suggestions.

Another Clinical QA system is introduced in paper [45]. It is named as Multi-source Integrated Platform for Answering Clinic Questions (MiPACQ). It accepts the free-format clinical questions and returns the answers based on its multiple knowledge sources. The word "integrated" in its name indicates that it combines both the IR and the NLP techniques together.

The basic steps involved in MiPACQ are: expected answer type annotation, questions semantic annotation, query term expansion, information retrieval, result set semantic annotation and answers re-ranking.

The researchers evaluated MiPACQ with a human-annotated evaluation database which is based on Medpedia health and medical encyclopedia. It has been observed and proved that MiPACQ performs 84% improvement in the precision value. Here, the evaluation indicator "precision" is defined as the proportion of the number of relevant documents retrieved by the system among the number of documents this system actually retrieved based on one test question.

Another significant factor in the medical information is the time factor. When did this record of a patient with the epidemic bronchitis stored into the system? When was the first case of the epidemic bronchitis being reported? The temporal information matters very much in the clinical research and the evidence-based applications. It determines how long a disease lasts, which season is a popular season for an allergic symptom, etc.

However, extracting and annotating the temporal information from the stored medical records are still the challenges for the current techniques. In some cases, the temporal information was not explicitly described in the original documents. It needs to be inferred in order to acquire a correct date. Not mention this time factor is also hard to be described in the search queries to retrieve the relevant and sophisticated documents in the database.

Based on this challenge, in paper [46] the writers come up with a clinic time-oriented QA system. They utilized a semantic web which is edited in Web Ontology Language (OWL) to offer a suitable environment for acquiring the temporal information. By accomplishing this goal, the researchers stored the temporal information into some related triples by using the clinic-domain ontology. The QA system then matches them to the related the queries and rank the candidate answers based on those time-oriented triples.

4. QA Systems on Mobile Devices

In paper [47], a new QA system whose name is Qme! is introduced and explained. Because of the strongly increased mobile device users, the novel feature of Qme! is that it is built on the mobile devices with a spoken language processing technique. Although there are still limits and disadvantages such as the small screens, inconvenient keyboards and weak signals sometimes, Qme! still successfully solves a part of the challenges by allowing the users use it outdoor without typing the questions by their hands.

Moreover, Qme! re-defines the questions' type by two terms: static and dynamic. The static questions are the questions that the system can retrieve the answers directly from the pre-generated answers. Apparently, the static questions can easily get the more accurate answers. For the dynamic questions, the system has to look for its relevant information from the related web pages. Unfortunately, the semantic web retrieval does not give back the better results to the dynamic questions.

5. Ontology Based QA System

In paper [48], the writers present their QA system which is named SemanticQA. It is a web-based QA system using the ontology as its main methodology. Especially, its ontology component is changeable with any other ontology for the different developing strategies. The writers also introduced their user interactive user interface in SemantiQA. When the users are typing their questions term by term, the system dynamically suggests some of the related worlds to the users to put into their questions. Those suggestions are made based on the entities' names the user just typed. For example, a user typed the word "professor" and the entity "professor" in the system's ontology database is connected with "faculty" and "department". Thus, these two words "faculty" and "department" will be shown under the question window immediately to help this user to form a more explicit question. In this way, the more formal questions with higher qualities are created by the cooperation between the users and the system. It helps the system to analyze the natural language questions and retrieve the better answers.

In paper [49], the writers introduce a close-domain QA system called AQUA. AQUA involves the technologies including the NLP, the ontology methodology, the logic interfering and the IR techniques in a uniformed frame work. The researchers took the great advantages of the ontology methodology to reform questions, map them with the relevant answers and measure the similarities between the questions and the candidate answers. Moreover, the answer similarity this QA system measures helps with the concept re-changing process for the better answers. It also helps with a logic formula to further evaluate those candidate answers.

Since the semantic web annotates the web resource with the semantic information, this type of annotations can be used by any other intelligent systems to perform a better semantic analysis. AQUA is then expected to be able to play a more important role in the process of annotating some home pages.

Sbuqa question answering system is another ontology-based QA system in paper [50]. The researchers designed this system with a Lexical Functional Grammar (LFG) for a deeper question analysis. The LFG is a meaning based grammar which analyzes the sentences at the semantic level instead of the syntactic parsing level. The difference between those two levels is that the semantic level contains more abstractive representations than a syntactic parser tree which only focuses on the relations between each entity. Therefore, the LFG is able to locate the topics of users' questions. Moreover, the LFG deals with the longer sentences which contain more details of the question topics. It is also capable to work with the multi-language situation and multilingual question answering systems.

6. User Interactive QA System

The QA systems are not only classified by the different software environments they are embedded into, they also can be distinguished from their different user interactive interfaces. In paper [51], the authors present a system named as NPCEditor. This open-domain QA system supports the users interacting function and creates the users' own assistances individually with the users' different preferences and demands.

The core algorithm of NPCEditor is a static text classification algorithm which helps the system mapping the questions to the related answers. It makes the system not over relying on the knowledge pre-training process of a huge amount of raw information. Thus means the new and random question which is a big challenge to the QA systems are no longer be replied by the low quality answers. The reason the writers are confident about this algorithm is the interactive dialogue the users use to communicate with the system. This strategy helps the system narrow down the relevant topics and the correct answers with more explicit details acquired from the user-system conversations.

Another paper [52] introduces two different user interfaces which allow the users interacting with the system. The clear and straight instructions are offered to the users during the interaction. Based on that, users can easily customize their own questions with some given semantic patterns. Therefore, the authors implemented their system with a Guide-Based User Interface (GBUI). Apparently, this system is not an open-domain system. It limits some certain knowledge areas so that the users can have a limited number of semantic patterns as the options to create their queries.

For the details of this GBUI, there are only 3 steps for the users to follow before they get the relevant answers back.

Step 1: Category Selection. As it was talked above, this system is a close-domain system. The users are only allowed to ask the questions in some certain fields. This step makes sure the users understand this feature and have their first chance to narrow down the question domain. Two levels of concepts are offered to the users to choose. For example, if users first choose the class of "natural", then they may choose the next level of concept as "animals" or "plants".

Step 2: Pattern Selection. Since the users have made their decisions about which area of knowledge they are interested about. Step two shows some question patterns they may use to form their questions. In this step, "What" and "How" question patterns are both listed. However, the users have only few options to choose.

Step 3: Question Formulation. In the last step, the question patterns the users picked up before are showing the main subjects the users can fill their own words in. For example: for the pattern "How long will [entity\event] last?" the users are allowed to fill one specific entity or event name in the square bracket to complete this question. Once they have finished filling with their own words, the system automatically formulates the question the users formed up and submits them to the system to search the related information.

This Guided-Based User Interface is certainly better than the traditional UI while working with a close-domain QA system. The users have a clear understand in each question forming step and they are sure about which type of questions they can ask to get an expected answer. A simple way to raise the satisfaction of the users' experience is that offering more question patterns in the second step above to allow the users have more options to customize their questions. However, it will also increase the complexity of the coming information retrieval and the total searching efforts.

7. Other QA Systems

7.1 Online Store QA System

As the online commercial websites have been developed and increased extraordinarily those days. The users who are seeking or shopping for some products online get more and more options than before. Thus, they have to compare the prices and the specifications of the products not only from one website but also throughout the whole internet. According to this fact, the researchers in paper [53] designed a close-domain QA system which helps the users acquire the best deal of the products they are looking for from all the B2C commercial websites.

Basically, this QA system uses two agents to accomplish this goal. The user negotiation agent deals with the natural language questions receive from the users. It analyzes the users' questions or demands, extracts the keywords and helps the users setting up the specific constraints for their ideal products. Then a web crawler as another agent crawls the necessary information from the targeted websites based on those key words and constraints. Finally, the user negotiation agent displays those useful and qualified product information based on users' requirements. For example, it displays the products from lowest price to highest ones to help the users finding the cheapest deal.

7.2 Academia Sinica QA System

The Authors in paper [54] represent a QA system called Academia Sinica QA system (ASQA). Five models were built up and used in ASQA. After the users submit their questions, the question processing model first analyzes the question by extracting its keywords and the topic. Those keywords and topic will help the document retrieval model to pick up some relevant documents throughout the corpus. However, as an intelligent QA system, feeding back just the original documents to the users is not the ultimate task. That is why the sentence selection model then retrieves and evaluates each sentence from those relevant documents. Only the qualified sentences whose evaluation marks are higher than a certain threshold will be ranked and sent back to the users as the candidate answers. Additionally, another model named redundant removal model is attached in ASQA as well. It detects those qualified sentences and checks if they are similar with others. Thus, the redundancy in those potential sentences will be eliminated and the system efficiency is then increased.

7.3 Music Knowledge QA System

For the close-domain QA systems, a special member of them is a music knowledge domain QA system [55]. It is an ontology-based system which deals with the music knowledge. It also works with a collection of Frequently Asked Questions (FAQ) for an efficient question process. This FAQ collection stores some popular music-domain questions and each question has already been answered. Those answers are also stored in the database along with their corresponding questions. The Music Knowledge Question Answering (MKQA) is composed with six basic models including the question classifier, FAQ question matcher, question analyzer and answers extraction. The users first type in their music-topic questions, those questions are then converted into eight general classes and forty-six children classes to match with the information stored in the FAQ. If the original question matches some questions in the FAQ collection, then the answers of matched questions in the FAQ collection are sent back to the users as the candidate answers without the further evaluations. If this is the first time MKQA system deals with some questions, it means those questions cannot be matched with the stored questions in the FAQ collection. Then the ontology analysis is executed to form some new answers to those new questions. Those new answers will be evaluated with some evaluation algorithms before being fed back to the users.

7.4 LogAnswer QA System

In paper [56], the writers introduce LogAnswer. LogAnswer is an open-domain QA system. It is different because of an automated theorem prover which restricts the candidate answers. This prover retrieves the answers from a logical knowledge representation by using the inference technology. It also considers the total time cost and the answer qualification for the whole system. Thus, the researchers set three main constraints for this prover to optimize its performance.

First of all, since LogAnbswer is an open-domain system, a qualified prover has to be able to deal with a large knowledge collection so that the prover can answer the questions on the wider topics. Secondly, since the prover controls the constraints of picking up the candidate answers, it must be also attached with a constraint relaxation loop to deal with the situation of no answer is retrieved. Another purpose of installing this relaxation loop is stopping and releasing the prover when it is processing some imperfect documents with an unaccepted time spent. The last requirement of the prover is supporting the answer extraction procedural. It needs to claim the answer substitutions to help the topic expansion for extracting the more relevant answers.

According to those three requirements, the writers designed two provers for LogAnswer system. One is called the MultiNet Prover which proves a question from a passage in less than 20ms on average. Another one is named as the E-KRHyper Prover. It participates in the reasoning procedure in LogAnswer.

7.5 Photo-based QA System

In paper [57], a very unique and experimental QA system is represented by the researchers. It is a photo-based QA system. For the users who use text-based QA system to ask questions about some subjects with the physical specifications, for example “How big is a 3kg watermelon”, they are expecting more than some words describing the 3kg watermelon. Therefore, the researchers came up with this photo-based QA system. By using the updated image matching technology, it is able to match a query photo with some photos online and acquire the internal semantic information from those online pictures. This semantic information will be fed back to the users along with those corresponding online pictures. For example, the users can upload a celebrity photo and ask who he is. The photo-based QA system will then match this photo with some similar photos online based on the question type “WHO”. It will give back a few celebrities’ names to the users by acquiring the text information from those matched photos. However, if we use an ordinary text-based QA system to deal with this users’ demand, the results will be empty since the users do not know any details of this person and it is hard to describe him or her with words. Even for some extreme cases where some photo-based questions cannot be answered, the researchers designed an expert community to answer some of those questions manually. This photo-based QA system is not only a novel multimedia information QA system; it also efficiently explores the usage of the online multimedia data resource.

7.6 Pythia QA System

In paper [58], a new QA system named Pythia is presented. The researchers who designed and implemented Pythia claimed that they had accomplished it mainly with two general ideas. First of all, they came up with a general meaning representation which can be converted into the formal queries for the system to understand. In order to conduct this conversion, they used the principle linguistic analysis.

The second idea their system uses is a new designed user interface which is based on the lexicon ontology. It means this UI requires the users to decide an explicit area of knowledge to help the system narrow down the number of retrieved documents.

7.7 Explanation QA System

In paper [59], the writers implement their theories with an open-domain QA system. The reasons that this system is different from others are: it feeds back the explanations of the candidate answers to the users as the references; its core theory is about the semantic graphs. First of all, the explanations of answers are comprised by three parts: a visual representation of documents, a fact list which is stored with the format as subject-verb-object corresponding to the documents, a summary for each document. Moreover, the fact of each document is created based on each sentence with the help of the Penn Treebank parse tree [60].

However, this system is not a complete open-domain system. The researchers who designed it claimed that not all of the question types can be understood by the system. To avoid this situation, they used a predetermined template to restrict the users' input. Furthermore, the documents in the database are stored with the semantic representations and graphs. The candidate answers are mapped based on those two data structures and the information on the document fact list.

7.8 True Knowledge QA System

In paper [61], an open-domain QA system called True Knowledge is created by the writers. Its database was built with both the static and dynamic knowledge with a structured knowledge representation. The natural language questions written by the users are converted into the language-independent queries. Then, the True Knowledge QA system uses these queries with the knowledge representation and a general interference model to map the candidate answers.

The knowledge representation helps the QA system analyze and store knowledge with an appropriate and efficient structure. For example, Enteritis 1: Big Ben, Relation: is located, Enteritis 2: London. This kind of structure performs a better understanding of the stored information for the relevant document retrievals.

7.9 Survey Research

In the survey [62], the writers compare some popular QA systems with their performances and some new updated technologies involved in this area. The writers did not include Google and MSN search engines due to their limited answering capabilities. Google only correctly answers some geography-topic question such as “how many provinces does Spain have”. Meanwhile, MSN search engine is only based on the Encyclopedia Encata as the knowledge source to answer the limited number of questions.

Therefore, the researchers have compared six QA systems and they have achieved some quite different results from each of them. The QA system called AskJeeves answers questions with its manually created database. Thus, the questions’ topics are also narrowed only by the encyclopedic requests. Ask.com which is an online open-domain QA system works with a keyword search engine called Teoma. Due to this fact, answering each test question with the up to two-hundred candidate answers does not offer a satisfied precision even though those answers are ranked for the users.

Moreover, only two of those six evaluated QA systems deal with multilingual languages. They are AskJeeves and AnswerBus. AskJeeves is also the only system which crawl the entire web. Others only acquire the knowledge from some particular websites or directly from some knowledge sources.

The writers explained those observations as the different purposes of the different QA system. The commercial QA systems are particularly interested in increasing their popularity for getting more advertising opportunities. The research-based QA systems which are not competing with the commercial systems are interested in exploring the newer and better algorithms on the academic purpose.

Summary

In this survey, the Question Answering (QA) systems and their involved technologies are introduced and summarized based on the latest sixty papers in this area. It is very clear that the QA technology is the most popular and developing technology recent years due to the high demand of the high quality answer searching experience. The new approaches made in this area are mainly following the trends which are Natural Language Processing (NLP), Semantic Information Crawling and Ontology methodology. However, the NLP technique is still the hottest topic among all of them. It is because understanding what the users are trying to ask is still the biggest challenge in QA systems. It is the key point to the QA systems to offer the high quality answers back to the users. Moreover, the popularity of NLP technique is also because it is highly needed in other areas which involve the user interactivities.

Furthermore, the Semantic Web Crawling has also been given more attentions than before since the questions are more needed being processed with the semantic patterns instead of the traditional syntactic patterns. Thus, these semantic-based queries need to be matched with the documents which contain more semantic information. In other words, the document collections should be pre-processed to be prepared for the semantic information matching. Thus, more correct answers are formed from the highly distributed passages.

The ontology methodology is another efficient choice for the document preparations. The relevant information is organized into the same class with the different level and also connected with others to represent its internal structure. This high structured database then raises the accuracy and efficiency of the whole performance of QA systems.

However, there is still not a perfect and mature strategy or system which is able to solve all the challenges and problems in this area. It leaves more spaces to the researchers to improve their approaches. We are looking forward to hearing more breaking news because they will change everyone’s life by answering their questions faster and better.

CHAPTER 2

My work

1. Introduction

As the Internet has become an important part of our lives. People are now more likely to search for the answers to their questions by using online search engines like Google, Bing, Yahoo, etc. However, the search results users get back are just links to web pages. Users have to look through a huge amount of links for the real answers they need.

In most cases, if a user types in their question as “What are the top five longest rivers in the world?” They will get about 5,890,000 links in 0.32 seconds. But all they need is a list of the names, locations and lengths of those five rivers.

During our initial research on the Google search engine, we used a number of “how-to” questions to test the quality of Google’s search results. Here are the results of our experiments: the average time Google used to retrieve and send back ranked links as the responses to the test questions was less than one second. Using Google the answers could be found among four of the top five links, on average. However, more links in the top five web pages may need to be followed by the users. More importantly, almost three of the top five links, on average, offered the direct answers.

Based on the results of our experiment above, we have designed a Question Answering (QA) system which improves the search results from Google by analyzing the searched web pages first and answering the users’ questions directly. Due to the high quality of the top links Google offers, the QA system uses Google as its search engine to perform an open-domain question answering service. It extracts text documents from the top links from Google, retrieves the relevant passages from these documents and sends the ranked passages as the candidate answers back to the users.

In the research area of the QA systems, they are defined as the systems which are able to automatically and directly answer users' questions. Researchers in the QA area have been trying to develop a system which can perfectly understand and answer users' questions submitted in natural language for the past fifty years. There are different kinds of QA systems which have different functions and features: biology QA systems; medical QA systems; interactive QA systems and so on. Those QA systems can also be classified into two general categories: closed-domain QA systems and open-domain QA systems. The closed-domain QA systems only answer questions in some certain topics such as the biology QA systems and the medical QA systems. The open-domain QA systems like our system, on the contrary, cover more general topics to answer more questions.

There are two main technologies involved into the QA systems: Natural Language Processing (NLP) and Information Retrieval (IR). NLP technology supports the systems for better understanding the users' questions. It usually contains the syntactic and semantic question analysis to properly convert the questions into the format that the machines understand. After acquiring the searching queries from the users, the IR technology makes sure the systems are able to retrieve the relevant and correct documents as the answers to the users' questions. Since the Google search engine has been comprised with a high-functioning NLP component for analyzing the search queries from the users, our QA system which improves the search results from Google mainly focuses on the IR improvements.

To generally introduce our QA system in this section, the system firstly offers a simple and friendly user interface to receive users' questions written in natural English. These submitted questions then are sent to the Google search engine by our system to acquire the corresponding Google search results including the ranked links of the related web pages.

Furthermore, a web crawler of our system analyzes the received Google search results and downloads those searched web pages into our local machine. Once the system receives those web pages, it will execute the document preparation. The first step of document preparation is to build a new document collection for the submitted question. Our system firstly detects the text contents on those downloaded web pages. The detected text contents will then be divided into several documents based on their original paragraph structure. Thus, each related web page is converted into several independent documents which contain one of its paragraphs.

The second step of the document preparation is indexing the entire document collection. Each term is indexed by our system with the document information of which it occurs. Moreover, the values of the Term Frequency (TF) and Inverse Document Frequency (IDF) are also calculated and attached to each indexed term. The TF value indicates the number of times a specific term appears in a particular document and the IDF value denotes how many documents this term occurs in the entire collection. Therefore, the term indexing helps the system not only locating the target words faster but also estimating how important they are individually in the different documents better. We use the Vector Space Model to build up an indexed term matrix for storing those term locations and their TF-IDF values for the document retrieval step coming next.

Our QA system has also been implemented with two strategies to increase the efficiency of the document indexing. The first one is about eliminating high-frequency terms such as “it”, “in”, “to”, etc. in the index. Those terms will disturb the document retrieval of our QA system by retrieving more irrelevant documents for the questions. Therefore, our system ignores those terms during the indexing based on a list of the high-frequency terms. The next strategy is called term stemming. To make sure the terms which share the same meaning but different formats are treated as the same word during the document retrieval, our system first stems each indexed term before it is saved into the index matrix. Thus, a group of terms such as “know”, “knew” and “known” will be indexed as the same term and detected in the document retrieval while one of them appears in the users’ questions. These two strategies involved into the document indexing makes sure our system to retrieve more relevant documents and less irrelevant ones.

The questions users submitted will be processed by the system as well. To be able to detect the semantic connection each time between the received question and one of the documents which do not share a common term in the collection, our system utilizes the searched web page snippets from the Google search engine to expanse the question. The question then is expanded by some high related terms offered by the snippets of the searched web pages. The expanded question will helps our system retrieve more relevant documents as the candidate answers to the users.

Based on the same high-frequency word list and the term stemming strategy used in the document indexing, the expanded questions are also processed before the document retrieval. The high-frequency words will be eliminated from the questions. For example, in the question mentioned above, “What are the top five longest rivers in the world?” will be processed and only “top five longest rivers world” will be left for the answer retrieval. The left terms will also be stemmed by the system to match more indexed and stemmed term in the document collection.

By using the indexed term matrix and the expanded questions from users, our QA system will then be able to match the relevant documents to the questions. It utilizes the traditional term-based document retrieval strategy to detect the common terms shared by the documents with the questions. Thus, as long as a document contains at least one common term with the expanded question, it will be retrieved back to the users as one of the candidate answers.

To rank the retrieved candidate answers for the users to view, we have also modified the measurement called Cosine Coefficient Similarity Measure (CCSM) to evaluate how relevant each candidate answer is to the question. The modifications we made are importing both the document length information and the resource web pages’ ranks into the CCSM based on the features of the documents in our collection. Since the documents our system deals with used to be the paragraphs in the web pages, the longer documents more likely contain the expected answers for the users. Also, the searched web pages’ ranks from the Google search engine offer some general ideas of the qualities of the text contents each of them contain. Thus, the documents extracted from the web pages which received higher ranks from Google probably contain the information with the higher quality. Overall, a retrieved document with a higher modified CCSM value has a higher possibility of containing the answer users are looking for. Our system then presented those ranked candidate answers in inverse order based on their modified CCSM values. Thus, the users are able to view the more potential documents first.

Our QA system also offers a novel step called question refinement as an advanced search to the users. After users go through the candidate answers for the first time, they will be asked if they have found the ideal answers they need in an accepted time. If their answers are no, there will be another search step asking users to refine their questions for the better results. The first step of this refinement is query reformulation. Users may find that some key words are missing and need to be added to the queries, or that some misleading words need to be eliminated. This decision is made by the users themselves based on the previews search result shown beside as the reference of their refinements. If the users create a new question, the system will send the new question to the Google search again to create a complete new document collection. The question revision changes the members of the candidate answers next round.

The second step of the query refinement is query weighting. Users will be asked to give a numeric value to each word in their queries to emphasize some key words in the next document retrieval. The higher value a word gets, the more important it will be treated during the next document ranking. The query term weighting affects the ranks of the future candidate answers.

The last step is about building a Boolean "NOT" query and bind it to the text question for a stronger restriction in the next document retrieval. The users can add some terms they do not expect to see in their future candidate answers. A document in the collection which contains some of the terms in the Boolean query will not be retrieved back no matter how many common terms it shares with the question. Same with the query revision, the Boolean query also has the power to change the constitution of the coming retrieved documents.

Thus, a new revised query which carries more useful information for the next document retrieval will be sent to the Google search engine again to get the new related web pages if needed. The whole process of document preparation, question processing, passage ranking and query refinement will be executed again until the users find the expected answers.

Overall, our QA system aims to utilize the results of the Google search engine and information retrieval technology to improve the quality of answers returned by the internet. In the following section 2, some related research works are discussed for the originality and improvements of our work. The sections 3 to 9 represent the details of each component of our system and the involved algorithms and strategies. The section 10 introduces the completely implemented system. We also discuss evaluation results of our improved algorithm and strategy in the section 11. The last section is the summary of our work.

2. Related Works

In this section, a novel document retrieval strategy is firstly introduced. The motivation, process details and advantages of this strategy are explained. The disadvantages are also discussed to indicate the differences between the original strategy and our improved retrieval strategy. Moreover, some related QA systems are also explained. We are discussing their specialties and shortages to emphasize the originality of our QA system.

2.1 Document Retrieval Strategy

Since our QA system first utilized the traditional term-based document retrieval strategy to match some relevant documents in the collection to the received users' question, only a document which shares at least one common term with a user's question can be retrieved by our system. For example, if a user type in a question as "what is AI", the document "Artificial intelligence is the intelligence of machines and the branch of computer science that aims to create it" will never be retrieved by our QA system. It is because this document shares no common term with the user's question. However, this document is highly related to the question and can be the best answer to help the user understand this concept. It should be retrieved and sent back to the user to be viewed. This is how we miss the valuable semantic connection between "AI" and "Artificial Intelligence" by simply using term-based document retrieval strategy to search for the related documents as the candidate answers. It clearly decreases the answer quality our QA system produces.

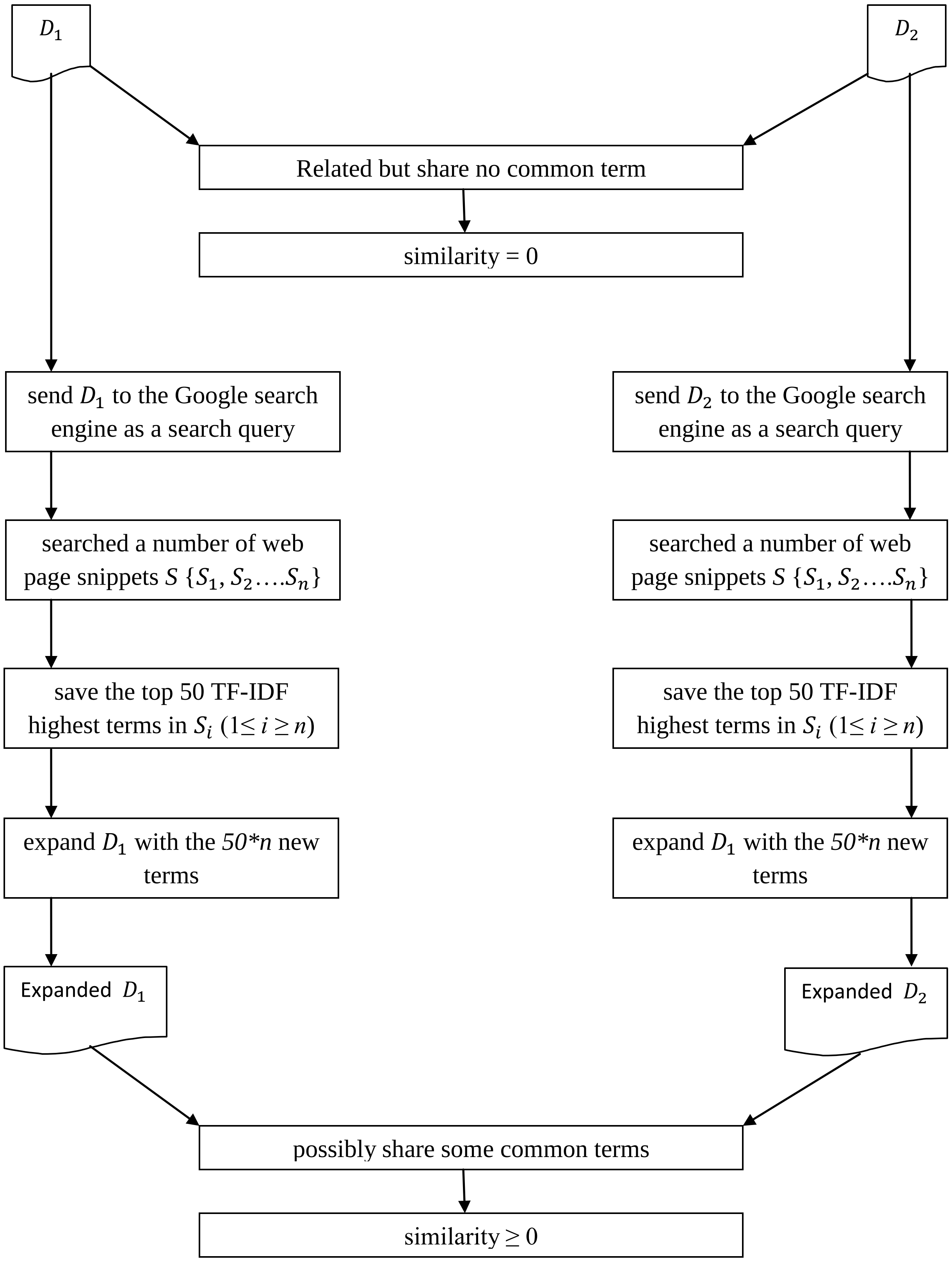
For avoiding this situation that some important and related documents are not retrieved by our QA system, we have made great efforts on researching and experimenting different algorithms and strategies. We have first utilized a traditional strategy called term stemming which helps our system detects more common terms without being distracted by the different term suffixes and prefixes. But we have also noticed that the term stemming still did not change the fact that the semantic information was missing during the retrieval. Because of the variety of information from the Internet and the unlimited questions we are dealing with. This was a big challenge to our system to dynamically detect the semantic connection between a question and one of the documents in the collection. Not only us, as it was talked in the chapter 1, other researchers in this area have also realized the affection of this fact and utilized different strategies such as the syntactic and semantic analysis on the document collection to fill the semantic gap between queries and documents. Among them, Mehran and Timothy [63] designed a novel and powerful strategy to try to solve this problem without integrating the systems with the complicated and expensive semantic analysis.

In their research, they claimed that in order to match two semantically connected documents together such as “AI” and “Artificial Intelligence” which do not share any common terms, a practical option is expanding each of them with some corresponding relevant terms. Apparently it has been proved that those two documents have a zero possibility of sharing common terms. With the two new expansions, they will definitely have a higher possibility to share one or more overlapping terms. Therefore, a QA system can combine the straightforward and affordable term-based document search strategy with the new document expansions to accomplish the retrieval without losing the semantic information.

Then how to expand two “irrelevant” documents became the most important topic in Mehran and Timothy’s research. They needed to carefully choose some qualified and related terms to expand each document in the collection with the consideration of the processing time. Clearly, using a collection of synonyms to attach the related terms to the documents is an option. However, the synonym collection can be out of date; some unimportant terms in the document have also been attached with their synonyms. Those disadvantages affect the quality of the expansions and the retrieved answer quality of the QA systems eventually.

Therefore, instead of using a synonym collection, Mehran and Timothy utilized the search results of the Google search engine. They first send the two “irrelevant” documents as two independent search queries to the Google search engine separately. Then, they will get back a few searched related web pages for each of the original document. The text contents on those web pages will be saved as the expansion resource of the corresponding original document for the next process. Apparently, simply using those text contents to expand the two original documents is not efficient. Each of the original documents will be expanded into a large text document. The system processing time will correspondingly be increased since more terms in each of the expanded documents will be checked during the document retrieval. The document collection of the QA system will be sharply expanded as well. Therefore, Mehran and Timothy first calculate the TF-IDF value for each term in the searched web pages from Google. Only the top 50 TF-IDF value highest terms in each searched web page will be saved to expand the original document. Thus, if a QA system uses the top 10 searched web pages from the Google search engine to expand a document in the collection, the original document will be expanded with 500 new relevant terms.

Moreover, Mehran and Timothy found out instead of using the complete searched web pages as the expanding materials, they could just use the web page snippets which are shown under the searched links on the Google search result page. When users use the Google search engine to search for some information, Google shows a web page snippet under each searched link to offer a quick view of each web page to the users. Those snippets are comprised by a few sentences which contain some common words with the search query. For example, the web page snippet of the first searched link corresponding to the query “what is AI” is “WHAT IS ARTIFICIAL INTELLIGENCE… This article for the layman answers basic questions about artificial intelligence. The opinions expressed here are not". Thus, if we follow the Mehran and Timothy's strategy to expand the document "what is AI", it will then be expanded with the terms such as "ARTIFICIAL" and "INTELLIGENCE". Therefore, after the expansion, the similarity between the two documents "what is AI" and "Artificial intelligence is the intelligence of machines and the branch of computer science that aims to create It." will become higher than zero as it was before. This is how the semantic connection between "AI" and "artificial intelligence" are detected based on the document expansions. Figure 1 further expansions the general Mehran and Timothy's expansion process.



D_1
D_2
Related but share no common term
similarity = 0
send D_1 to the Google search engine as a search query
send D_2 to the Google search engine as a search query
searched a number of web page snippets S $\{S_1, S_2 \ldots . S_n\}$
searched a number of web page snippets S $\{S_1, S_2 \ldots . S_n\}$
save the top 50 TF-IDF highest terms in S_i $(1 \leq i \geq n)$
save the top 50 TF-IDF highest terms in S_i $(1 \leq i \geq n)$
expand D_1 with the *50*n* new terms
expand D_1 with the *50*n* new terms
Expanded D_1
Expanded D_2
possibly share some common terms
similarity ≥ 0

Figure 1

Since the Google search engine offers a better search result with a smaller text query, each document we send to Google to acquire some relevant web page snippets for the future expansion should be small as well. This was the reason Mehran and Timothy claimed in their paper that their retrieval strategy only works for the short queries and the small documents. However, this fact does not affect the testing result of their work. Based on their evaluations, this snippet search strategy remarkably raises the similarity between the traditionally “irrelevant” documents. Comparing with using a synonym collection as the expansion material, the web page snippets help the system detect more semantic connections between two sentences or two small passages. For example: the new search strategy is able to show out the semantic connection between “java programming” and “applet development” by the helps of their corresponding web page snippets from the Google search engine. But the experiment showed that using the synonym collection failed in this case.

Mehran and Timothy also explained in their paper that their search strategy is suitable for the query suggestion systems. As a query suggestion system, it offers users some similar and more valuable queries based on users’ own queries to assistant them acquiring the information they are looking for. For example, if a user type in a question “which laptop should I buy”, the query suggestion system will then show the user some similar but more specific queries such as “2012 best selling laptops” to suggest the user to use. Hopefully, the new suggested query will offer the user a better search result than the user’s original question. In the query suggestion system which is using Mehran and Timothy’s expansion strategy, there is a local pre-processed query collection in which each query has been expanded by their web page snippets and stored. Thus, more relevant prepared queries can be matched to the users' question as the suggestions. Every time a user type in a new query, the query suggestion system sends only this piece of original query to the Google search engine to get expanded. The new expanded query will then be matched with some of the prepared queries in the local collection. Even though the pre-processed query collection may be large, each query of this collection needs only one time expansion. The future query expansion will only make on the received users' questions. Therefore, Mehran and Timothy defined their new snippet search strategy as a lazy strategy which means the document expansions only be executed when they are necessary.

However, even though Mehran and Timothy claimed that their snippet search strategy works with different term-based similarity measurements, it is not practical and efficient to directly embed this strategy into some QA systems which perform the term-based document retrieval on some dynamically updated document collections like our QA system does. Those QA systems initialize a complete new document collection based on the new question they receive each time. In our QA system, each time a new question is submitted from the users, the system sends the question to the Google search engine to search for the top 10 web pages back. Then a new document collection for answering this question is set up by distributing each text paragraph in each searched web page and storing them as independent documents in the collection. This means if we directly utilize Mehran and Timothy's search strategy to detect the semantic connections between the received query and the corresponding document collection which usually owns more than 200 documents, we have to send more than 200 text queries to the Google search engine to expand the documents and the users' question with more than 2000 web page snippets.

Our QA system clearly cannot afford this expensive search strategy. In the section 7, we will explain how we have modified and improved Mehran and Timothy's strategy to embed it into the dynamic term-based QA systems like ours. We used their strategy as the basic concept and inspiration to develop a new document search strategy.

2.2 Related QA Systems

Moreover, our QA system is not the first QA system which keeps the access to the information from the internet to answer users' natural language questions on the open topics. Other researchers have also noticed the great value of the information online. Comparing with the building up a knowledge database by acquiring the necessary information from some particular existing knowledge resources, using various online web pages to extract the useful knowledge for answering users' questions certainly has a few advantages. First of all, the information from the Internet enables the QA systems answering more questions on more topics. Especially for the open-domain QA systems, the topics of the users' questions are not limited. To guarantee this service, the open-domain QA systems need to have more general knowledge databases which cover more topics. Certainly, there are no other information sources better than the web pages on the Internet to supply this demand of those QA systems. Secondly, due to the dynamism of the Internet, the online information the QA systems collect is more up to date. The QA systems then are able to answer some users' questions with the information extracted from the daily news and newest research approaches. The last, the users some time more prefer being answered by some informal but experiential answers which are usually from some online forums where discussions and communications are made. For example, the query "Need some Italy travel tips" expects some answers from the people who traveled in Italy before, professional and formal information such as “Rome is the capital of Italy”, “The population of Italy is 60,626,442” is not the ideal answer to this question. Thus, the QA systems which acquire its knowledge from the Internet are more capable to answer the users’ unlimited both casual and personal questions.

As one of the open-domain QA systems which connect to the Internet talked above, Valentin and Maarten [64] represent their QA system which has been developed to answer users’ natural language questions on the unlimited topics. This system concentrates on extracting the knowledge from a particular kind of web pages: Frequently Asked Questions (FAQ) web pages. On the Internet, there are a huge number of questions which have been frequently asked by the Internet users. The FAQ web pages collect those high-frequency questions with their best answers and show the pairs of questions and answers to the users for saving their time asking the same questions again online. Valentin and Maarten noticed this huge value of the FAQ web pages and decided to utilize it as the knowledge resource of their own QA system. They first visited a number of the FAQ web pages and properly analyzed those web pages so that they can extract the pairs of answers and questions correctly. Those downloaded pairs of questions and answers then were saved into the data base together to complete the knowledge collection of the system. Once a user asks a question to this QA system, the stored high-frequency questions will be compared with this received question. The similar stored questions then will be matched and ranked based on the similarities between the question and themselves. At last, those marched high-frequency questions will be sent back to users with their corresponding answers as the potential answers to the user’s question. Based on Valentin and Maarten’s evaluation of their QA system, 3.6 GB of text data have been downloaded from 293,000 FAQ web pages. By using this huge question-answer database, this QA system successfully answered 36% of tested questions with the top 20 candidate answers.

However, the variety of the questions' topics on the FAQ web pages is still limited. Only the questions which have been asked frequently will be able to be collected on the web page and downloaded into Valentin and Maarten's QA system eventually. This fact decreases the ability of their QA system to answer users' open-domain questions. Thus, it explains the evaluation result that more than half of the tested questions were not answered. However, as it was explained in the introduction, our QA system is dynamically connected with the Google search engine. Each time our QA system receives a new question from a user, it sends the question to the Google search engine to get the different related web pages for the future retrieval. It means not only the topics of users' questions but also the retrieved answers are not limited by our system. As long as the Internet contains uses' expected answers, our QA system is able to detect them and represent them back to the users. This is the major difference between Valentin and Maarten's system and ours. This difference leads the fact that our system can answer all the questions Valentin and Maarten's system answered since we also have the access to the FAQ web pages by the help of the Google search engine. Our QA system also covers more questions' topics than Valentin and Maarten's system does.

Another related work in the QA area was made by Jimmy and Boris [65]. They designed and implemented a QA system called Aranea. Aranea is a web-based QA system. Jimmy and Boris claimed and explained in their paper that due to the huge amount of information on the Internet, accessing a web search engine to retrieve the potential answers works better than having a local corpus to answer users' questions. Thus, we share the same system designing motivation with Jimmy and Boris. Moreover, they also explained in their paper that due to the special feature of the Internet: data redundancy [66], only one retrieved candidate answer which was extracted from the Internet may not fully answer a received question. A group of related candidate answers offers a better result. Therefore, this theory supports our QA system's search strategy from another side as well. Our QA system answers the users' question with a list of ranked passages. Those retrieved passages are originally from the different web pages of the Internet. Based on Jimmy and Boris' theory, this list of ranked passages performs a more complete answer to the users' question.

However, there are some major differences between Aranea and our QA system. First of all, Jimmy and Boris used two different strategies to prepare their dynamic document collection for each of the users' queries. Unlike our QA system, Aranea saves the text content on each searched web page as an independent document after it sends the users' question to a web search engine as a search query. However, these searched web pages in our QA system are just treated as the intermediate materials. Our QA system distributes each text paragraph on each web page and stores them separately in our document collection. Thus, our document collection contains the smaller independent passages which use to be the text paragraphs in the web pages. The two document collections are prepared with two different ways.

Secondly, after finishing preparing its document collection, Aranea utilizes the basic and traditional pattern matching strategy to map some related sentences in the downloaded documents to the query. For example, to retrieve the candidate answers to the question “who invented the telephone”, Aranea extracts the query pattern “invented the telephone” from the question and matches the similar sentences in the document collection. Therefore, only the sentences which contain the exact same pattern with the query will be retrieved. Some valuable sentences such as “Alexander Graham Bell and Thomas A. Watson successful made the experiments with the fist electronic telephone device in the late 19th century” are missed during the retrieval. However, our QA system has been integrated with the term-based document search strategy to retrieve the potential answers to the questions. It means, with the help of this search strategy, documents are able to be searched by our QA system as long as they share the common terms with the query. To the same query and document exampled above, since this document share the common term “telephone” with the query, our QA system is able to retrieved it and send back to the user as one of the candidate answers. In other words, the term-based document retrieval strategy helps our system break the limitation of the traditional pattern search strategy. No existing solid pattern which stops us retrieving the potential answers with various formats is used into our search strategy.

Moreover, as it was explained at the beginning of this section, we have modified and embedded a snippet search strategy into our QA system. This improved strategy allows our QA system to detect the semantic connection between the query and each document which does not share the common terms with the query. It remarkably raises the answer quality of our QA system. However, like other traditional QA systems, Aranea misses the semantic information during the answer retrieval. The original users’ questions are the only references Aranea has to search the potential answers. Apparently, the implicit information in the document collection is missed.

To the best of our knowledge, our QA system is a complete and unique QA system in this research area. Its originality is guaranteed by its concentration and the integrations of carefully improved retrieval and ranking strategies. The coming up sections will fully explain the algorithms and performance of this system to further prove this fact.

3. System Diagram

Before going further to the details of each component and strategy integrated in our QA system, the system diagram should be first introduced to offer the general ideas of what the input and output of our QA system are; how different components are cooperating and working together to produce the expected results. This section will help us explain the each system processing steps in the following sections by showing a clear system structure first.

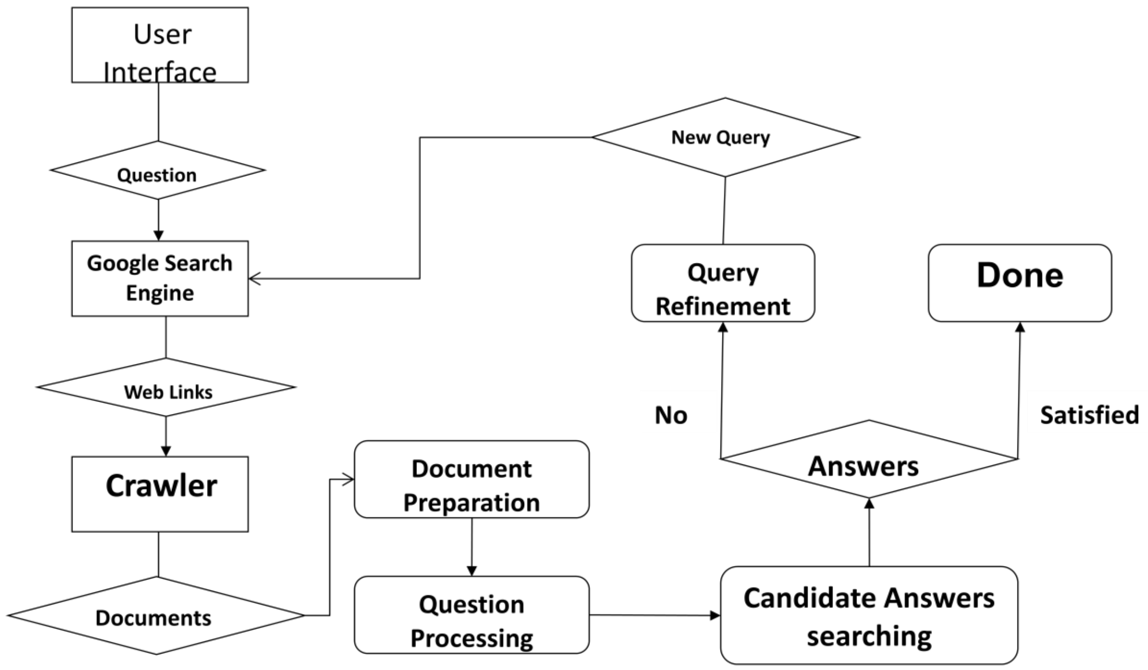


Figure 2

In Figure 2, the complete process of the answer retrieval is described. The independent objects involved into the retrieval are in the rectangles. The diamonds indicate the system intermediate and final text productions. All executed steps are presented in the rounded rectangles. Arrows in Figure 2 denote the directions of the data flow.

The first main task of our QA system is building a knowledge resource which is a document collection for each of the users' question. According to Figure 2, the user interface of our QA system firstly receives a text question written in natural English from a user. At this point, our QA system does not pre-process this piece of question. The system simply saves the question and directly sends it to the Google search engine. It is because the Google search engine does the query processing itself before searching the relevant web pages. After receive the question, the Google search engine uses it as a search query to retrieve the top 10 to 20 related web pages. It then sends those searched and ranked relevant web page links back to our QA system. Those web page links are exactly the same links the Google search engine shows to other Google users who type in the same question. Furthermore, those links are sent back in a text package which has its own data structure to our system. In this text package, there is other information of the related web pages such as: the titles, web page snippets, web page types, etc. A web crawler of our QA system then will be executed to extract the pure web links from this package and use them to visit the searched web pages. After the link extraction, the crawler visits each of the searched web pages, detects their text contents and downloads them into the local machine. However, the web crawler does not just simply save the complete text content on each web page as a text document. It analyzes the natural paragraph structure of the text content and saves each paragraph separately into our initialized document collection. Thus, our paragraph-based document collection is created exclusively based this received question.

Before our QA system starts to retrieve the relevant documents as the candidate answers to the question. There are two major steps need to be followed to pre-process the local document collection and users' query for a better search result. The first step is the document preparation. The paragraphs which have been separately stored in the document collection are first named with some identical digits to indicate not only their identities but also the source web pages they were extracted from. The reason of naming them in this way is that the Google search engine sends us a list of ranked web links to denote the quality of each web page. The higher rank a web link got, the better information this web page stores. Our QA system should not miss this important information during the next answer retrieval. Therefore, in this step, the system numbers each web page link with its own rank. For example, the top 1 web page link gets the number of 1 showing it is the best web page based on the Google's opinion. Then all the paragraphs extracted from this web page get the suffix of "1" in their own identical file names. Thus, our document collection classifies different documents based on their source web pages.

However, this is not the whole process of the document preparation. More valuable information will be analyzed and extracted from the documents in the next step of document preparation: document indexing. For each document, each term it contains is detected and evaluated during this step. The detection helps our system save the locations of different terms from the different documents. Those term locations compose the index of this document collection. With the help of this index, the system is able to search for a particular document or term faster and more accurate. Also, during the document indexing, a list of words such as “is”, “in”, “at”, etc. which are named as stop words is used. The stop words on this list are the words occur frequently in the general English documents. Those terms do not help our system distinguish each document but decrease the indexing efficiency and retrieval accuracy eventually. Thus, those high-frequency terms are not indexed by our system. Moreover, our QA system evaluates how important each term is in each document and saves the term evaluation results along with the term location in the index. This evaluation involves a few of term-based algorithms which will be explained in the section 6. The term evaluation result is that each term in each document is assigned with a numeric value indicates how important it is not only in a specific document but also in the whole collection. With these values, our QA system is able to “understand” the documents and find the more relevant ones based on the users’ questions.

To decrease the sensitivity of the term-based document retrieval strategy our system is going to use to retrieve the related documents for the users’ questions, we also imported the term stemming strategy into our document indexing process. The term-based document retrieval our system conducts only match the exact same terms in the documents to the users’ question. Those documents which contain the searched common terms then will be considered as the related documents with the questions. Thus, fewer related documents will be searched due to the sensitivity of this term matching process. The imported term stemming strategy strips each term of their prefixes and suffixes before indexing them. Therefore, terms which share the same stems will be treated as the same terms during the indexing. Our system then is able to match the common terms without being disturbed by the term tenses, numbers and different formats.

Another preparation before the document retrieval is the question processing. Our QA system simply detects and eliminates the high-frequency terms in the users' questions based on the same list of stop words used in the document indexing. This step did not executed at the first time when our system receives users' questions because the Google search engine also deletes the high-frequency words in the questions. Thus, our local stop-word elimination does not affect the Google search results. For matching with the documents which have been indexed with their stemmed terms, each term in the users' question will also be stemmed in the same way with the terms in the document collection. After those two steps, the users' questions will become the "pure" search queries and ready to be answered.

At this point, our QA system has the indexed and evaluated documents in its collection and the "pure" question. Before it is ready to match the relevant documents to the query, one more important step is needed. Our QA system needs to extract the snippets of the searched related web pages in the text package the Google search engine sent back. As it was mentioned in the related works and will be completely discussed in the section 7, the web page snippets describe the connection between the search query and the web pages. They explain the reason why those web pages are searched to the query by the Google search engine. In other words, these snippets are highly related with the query which is the users' question in our case. Our QA system utilizes this special characteristic of the snippets to expand our received question before the answer retrieval. With the help of this query expansion, our QA system is able to detect more semantic connections between the documents and the questions.

Following up is the core step of the system: the answer retrieval. Based on the important preparations made above, our system is able to go through each document in the collection and match the related ones to the expanded query. As long as a document shares a common term with the expanded query, it will be searched by our QA system. Furthermore, the searched documents are not sent back to the users directly as the candidate answers to be viewed. To reduce the users' efforts searching for the best answers among the entire searched documents, our QA system ranks the candidate answers based on how similar they are individually with the question and presents them in the descending order based on their similarities. Thus, lists of retrieved and ranked candidate answers are presented to the users to answer their corresponding questions.

This last step in Figure 2 is the query refinement. It is an optional step which only depends on users' opinions of the candidate answers they just received. Users will be asked whether the retrieved documents have answered their questions or not. If their answers are no, the option of the query refinement will be offered to them to help them modify their questions. In the query refinement, users need to follow the three steps to optimize their queries. The first step is reediting their questions by adding or deleting terms if necessary. Secondly, they can also assign different numeric values to each of the terms in their questions to emphasize some key words. If a term in the question gets the highest value among others, this term will be treated as the key word during the next answer retrieval. The last refinement is combining the existing term-based query with a Boolean query. Users have this opportunity to bind a Boolean "NOT" query to their text query. Any words assigned to the “NOT” query will lead the result that no documents which contain those particular terms are shown in the next search result. Thus, this Boolean query offers a strong restriction to narrow down the candidate answers next time. Moreover, those three refinement steps are optional to the users to choose. If a user skips the first step reediting the question, our QA system will not send the same question to the Google search engine but use the same document collection during the next answer retrieval. If the user changes the terms in the query, the new query will be sent to the Google search engine to retrieve the new relevant web pages as the materials for building up a completely new document collection. As it has been seen in Figure 2, the whole answer retrieval circulation will be executed again and again until the users are satisfied with the answers they received last time.

4. Google Search Engine

Since our ultimate goal is improving the retrieval of information from the Internet, our QA system uses the web pages on the Internet as our original information resource to process. On the other hand, to build up an open-domain QA system, we do need to have an access to a large knowledge source where the information on different topics is stored. By the help of the variety of the large knowledge source, our QA system will be able to deal with the questions in unlimited areas and answer them properly. Based on this system demand, the Internet is clearly the best choice for us. There are no other sources or databases work better than the Internet to offer our system the unlimited and up-to-date knowledge to process and retrieve. Moreover, the Internet contains the information in not only different domains but also different types. Besides the professional and formal knowledge such as the research articles and daily news, there is also the casual but experiential information which usually can be found in the online forums and question and answer communities. Thus, if a user asks the question “can you show me some Spain travel tips”, our system is able to answer it with the information extracted from some Spain travel forums where other tourists share their experiences and recommendations at. Therefore, those are the two other main reasons we chose the Internet as the knowledge resource of our QA system.

At the system designing phase, we had two options to design our system: a static system or a dynamic system. In our research, the static QA system is defined as the system which is attached with a solid and pre-process database. It largely predicts and imports the knowledge before it receives the questions from the users and may update its database occasionally. In most cases, the static QA systems are the close-domain QA systems. It is because they have to limit the topics of the users’ questions so that they can import the useful information in those limited areas for only once while they are preparing their knowledge databases. In other words, they have to make sure they only receive the questions which can be more likely answered by using their prepared database. The advantage of the closed-domain QA systems is that they can be better prepared for the questions since the knowledge are only needed from the limited areas. On the contrary, the dynamic QA systems do not fully prepare their knowledge databases before receiving the questions. They only import the knowledge purposefully when they know which field of information they need to answer a specific question. In other words, the dynamic QA systems treat the questions individually as the references of their knowledge importation. Thus, some of the open-domain QA systems can be classified into the dynamic QA system category. Based on our system demand discussed above, it was an easy decision to make to build our QA system as an Internet-connected dynamic open-domain QA system. By following this system designing direction, our system was designed to be able to answer users' questions in different and unlimited areas with the appropriate and up-to-date information from the Internet.

In order to keep a permanent access to the Internet and download the necessary information dynamically, our QA system needs to choose and connect to a web page search engine. There are three main facts to evaluate a web page search engine: coverage, efficiency and accuracy. An ideal web page search engine which reaches all the requirement of our QA system has to be able to visit all the web pages on the Internet; search an great amount of the pages in an acceptable time and retrieve only the relevant ones based on the search queries. Therefore, according to Appendix A and Appendix B, the Google search engine was the best choice for our system. Connecting to the Google search engine is helping us not only retrieve the huge amount of information online, but also analyze the users' natural language questions without implementing the expensive and sophisticated NLP techniques. As the matter of fact, the Google search engine offers users a free access to connect with itself. Users can acquire the search results by sending the text queries to the Google search engine. By the help of this service, our QA system has been successfully connected with the Google search engine and utilizing its search results as our knowledge resource.

The information communications between our system and the Google search engine are based on the search queries and text packages. Each time our QA system receives a text question from a user, it saves the original question in the local machine and send the question to the Google search engine as a piece of search query without any further modifications. The Google search engine then will receive the query and search for the relevant web pages on the Internet based on it. The search result of Google is a list of searched links which are the addresses of the relevant web pages. After the Google search engine successfully produces this search result, it will comprise the result in a text package with a specific data structure. In this text package, there are not only the web page links the Google search engine searched, but also the valuable information such as: the titles of the searched web pages, the web page snippets which describe the connection between the search query and the corresponding web page, the total time the search process cost, etc. Therefore, we have designed a special web crawler in our QA system which is used for detecting and extracting the web page links from the text package. After the detection and extraction, the web crawler visits each searched web page and downloads the necessary information into our local machine. Moreover, those links are not the only information our system needs in the text package from the Google search engine. It also needs the web page snippets mentioned before. The web crawler also extracts and saves them into our system. The purpose of this step will be explained later in the section 7. So far, the web page links and snippets have been saved into our system for the answer retrieval.

Furthermore, the main tasks of this web crawler of our QA system are visiting each corresponding web page based on the list of links it just saved and extracting the valuable and useful text information from them. Since the text contents on the web pages are mostly the main components on the Internet, our system only processes the text information as the knowledge resource to the users' question. There are also a few facts affect the web crawler downloading the text information. Firstly, as it will be fully explained in the following sections, our QA system is going to use the term-based document retrieval strategy to search for the potential answers in the document collection. This strategy only detects and determines which independent documents are qualified to answer the received question. Our system does not further extract a specific sentence from a particular document as a candidate answers.

Moreover, as long as an independent document shares at least one common term with the question, this document will be matched to the question and completely presented to the users. Thus, the retrieved documents determine the readings the users have to make to find the expected answer. Thus, there were considerations about the size of each individual document in the collection. If the web crawler simply saves the searched web pages into our document collection completely as the separate text documents, since those pages have been approved by the Google search engine that they share the common words with the users' question, they will all be matched by our QA system's answer retrieval strategy and sent back to the users. Then there is no difference of the search results to the users between our QA system and the Google search engine. In other words, no improvement would have been made by our system. On the other hand, if the web crawler exacts each sentence from the searched web pages and store them separately as the independent documents, some important sentences which do not contain the common terms with the question may not be retrieved by our QA system. This is not acceptable neither since missing those sentences decreases the answer qualities of our QA system.

Therefore, comparing with these two extreme cases above, our QA system makes a compromise between those two document downloading strategies: it saves the text information from the web pages based on their natural paragraph structures. Thus, the text paragraphs on the web pages are distributed and stored separately into our collection as some independent documents. Since only a few of paragraphs will be sent back as the candidate answers, our system then has successfully narrowed down the useful information for the users and improved the retrieval results from the Google search engine. Because of the matched paragraphs also carry some other sentences which do not share the common terms with the questions but may be valuable, our system can also present an acceptable amount of extra related information along with the matched documents to expand and complete the potential answers.

While the different paragraphs are downloading from the web pages, an efficient strategy to store them into our document collection is needed. First of all, as a dynamic QA system, our system initializes a complete new document collection each time it receives a new questions in order to keep the knowledge related and up to date. Therefore, before the new paragraphs are saved into the collection, the previews stored documents will be eliminated completely to initialize the whole document collection. Moreover, to identify the paragraphs correctly, the system names each text document with five digits exclusively. The first two digits specify the web page this document was extracted from. The Google search engine has ranked those searched web pages based on their relevance with the query. Theoretically, a higher rank a web page got, a bigger possibility this web page contains the correct answers. Thus, the ranks of the web pages denote the quality of their paragraphs. Therefore, our QA system uses the first two digits of each document's name to store the rank information of its corresponding web page. During the future candidate answer ranking, these ranks will be considered as the references to evaluate the documents. The last three digits in the documents' names are the identical information to distinguish them individually. Thus, the dynamic document collection has been carefully built up and ready for the next process.

5. Document Indexing

After all the text contents from the searched web pages have been downloaded and stored properly into the document collection by our QA system, there are a few more steps to prepare the documents before the system starts to match some of them to the received question. The first step is to completely index all the documents. As our QA system deals with a text-based document collection, it needs to locate each term in the documents correctly and efficiently in order to match the common terms between the question and the documents. This is the reason our QA system stores each identical term in the index with the documents' IDs of which the term occurs. For example, if the term "mobile" appears in the document number 01011 and 05059, our QA system will attach those two document IDs to the word "mobile" in the index. In the answer retrieval coming next, if the term "mobile" is also found in the received question, our QA system then is able to match the document 01011 and 05059 to this question since they are related according to this common term.

However, not every single term in the document collection should be indexed. The high-frequency terms in the documents such as "in", "to", "at" do not need to be matched to the question since they do not represent the semantic meaning of each document and the question. Using those high-frequency terms in the document indexing and retrieval will decrease the quality of the candidate answers eventually. It is because more irrelevant documents which share some common but useless terms with the questions are retrieved by our system as well. Therefore, during the document indexing, our QA system ignores all the high-frequency terms based on a list of stop words [67]. The words on the list will not be recorded with their occurrences so that the system will no check them during the common term matching. Based on the same purpose, our system also eliminates all the high-frequency terms in the users' questions. Thus, no such terms appear and disturb the document retrieval.

There is also another retrieval situation our QA system should avoid of. When a user ask the question "who sponsored the first Olympic Game", the document "Evangelos Zappas, a wealthy Greek-Romanian philanthropist, was the sponsor of this event in 1856" which is the perfect answer to this question will not be retrieved. It is because this document does not share a common term with the question. To our QA system which indexes its document collection by using the spelling to identify each word, the terms "sponsor" and "sponsored" as two completely different terms to our system. Thus, the groups of English terms which share the common stems are not treated as common terms. It leads us to the result that valuable documents are missed during the answer retrieval like the situation mentioned above. To avoid it, our QA system conducts a process called term stemming. During the document indexing, while the system are recognizing and recording the each term, the term it is dealing with firstly is stripped of its suffix and prefix in order to be saved only with its stem. For example, the terms "sponsor", "sponsored", "sponsorship" will be stemmed and only the stem "spons" will be stored. The documents all those three terms appear will be recorded under the stem "sponsor". Thus, our QA system will treats those three terms as the same term during the next common term matching. Same with the document term stemming, after the system receives a question from a user, each term in the question besides the high-frequency terms will be stemmed as well. Thus, the question and documents from the example talked above will be match together based on the stemmed term “sponsor”. Therefore, the term stemming strategy helps our QA system match more relevant answers.

This document indexing strategy makes sure our QA system locate the related documents for the users’ questions correctly and efficiently. The involved high-frequency word elimination and term stemming help the document retrieval match less irrelevant passages and more relevant candidate answers respectively.

6. Term Weighting

If more than one relevant document is retrieved by our QA system based on the received question, the system ranks those candidate answers in descending order based on their similarities to the question. Since the documents with higher ranks carry more valuable information, the ranking will save the users’ efforts of searching the expected answers among the entire retrieved documents. The document similarities are the standard to evaluate how similar a searched document is to the question. The document which is more similar with the received question is more likely containing the expected answer. It then should get higher rank among the retrieved documents so that it will be viewed by the users earlier. However, the number of common terms each retrieved document shares with the question is not the only information we use to measure the document similarity. Our system also considers how important each matched common term is in both the question and a specific document to determine how relevant this document is to the question. We define the term importance here as the term weight and calculate this factor for the each term during the indexing.

Since each document in the collection is comprised by a number of terms, those terms then express the ideas and the details of the document they belong to. Therefore, understanding those terms helps us determine which documents are similar and how similar to the users' questions. In other words, they affect the similarity between the documents they are contained by and the questions users asked. The way our QA system understands each document during the document indexing is firstly calculating the number of times each term occurs in a particular document. For example, if the term "lion" occurs 5 times in the document 06087 and 2 times in the document 03027, then to our QA system, the first document carries more information about lion than the second document. Furthermore, if a user asks the question "where do lions live", the first document will be considered as more relevant to this question than the second one. It will certainly be ranked higher based on this fact. Therefore, the Term Frequency (TF) [68] of each term in the different documents is one of the facts our QA system considers to understand the documents and determine how relevant they are with the users' questions.

There is another factor term our system considers during the term weighting. It is called Inverse Document Frequency (IDF) [68]. Here is the formula of IDF:

$$IDF = \log \frac{N}{n}$$

In this formula, the N represents the total number of documents the collection contains. The n indicates the number of documents a specific term occurs. Therefore, as long as our system has built up a document collection based on a particular received question, the N becomes a constant number. The only variable in this IDF formula is the n which depends on the term it represents for. For example, in a document collection which contains 500 independent documents, if the term “lion” occurs in 233 documents, the IDF value of the term “lion” will be around 0.3316 based on the formula. Thus, the IDF value indicates only the document occurrences of a term in the whole collection. If the term “lion” occurs multiple times in one document, the IDF formula still only treats this fact as one time occurrence in that document. Therefore, the IDF value offers us a different and wider view of how important a specific term is in our document collection.

Furthermore, there is the major difference between the TF value and the IDF value. a term has a higher TF value other terms’ in a specific document when it occurs more times than others terms in this document. On the contrary, since the variable n is the denominator of the constant N in the IDF formula, the IDF value goes lower if the corresponding term occurs in more documents in the collection than others. Thus, the variable n and the IDF value share an inverse ratio growth.

Therefore, our system utilizes the combination of the TF and IDF weighting formulas which is called TF-IDF [68] to evaluate each term in the different document. The TD-IDF weighting strategy helps the system understand how important a term is in not only one particular document but also the whole collection. Here is the formula of the TF-IDF weighting strategy:

$$TF\text{-}IDF = TF * IDF$$

Based on this formula, a term in a particular document gets a higher TF-IDF value means it occurs frequently in this document but not frequently in the whole collection. For the example talked above, the term “lion” occurs in 233 documents of a collection which contains 500 documents in toal and this term appears17 times in the particular document

05100, then the TF-IDF value of this term in the document 05100 is 17 * 0.3316 which is about 5.6372. Apparently, this TF-IDF value only indicates the weight of this specific term in this particular document. The same term in different documents may have different TF-IDF values.

While Our QA system is indexing the document collection, it is evaluating each term in the different documents with the TF-IDF weighting strategy. Therefore, two kinds of information of each term are extracted from the documents during the indexing: the term location and the term weight. As it was explained before, the term stemming strategy and the high-frequency term elimination are also used during this process. Terms with the same stem are located and weighted as the same term and the high-frequency words on the list are ignored.

To store the term locations and weights together for the coming process, our QA system uses the Vector Space Model (VSM) [68] as the matrix to save the index results. As it is shown in Figure 3, the x coordinate axis represents the identical terms in the whole document collection and the y coordinate axis specifies the document IDs. Each term's weight in each document is saved as the value of each tuple in the matrix. For example, the w_{nm} is the TF-IDF weight of term *n* in the document *m*.

$$
\left(\begin{array}{cccc}
 & T_1 & T_2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & T_n \\
D_1 & W_{12} & W_{22} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & W_{n2} \\
D_2 & W_{11} & W_{21} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & W_{n1} \\
\vdots & \vdots & \vdots & \vdots \\
D_m & W_{1m} & W_{2m} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & W_{nm}
\end{array}\right)
$$

Figure 3

Finishing building up this indexed term matrix means this end of the document preparation. Thus, our QA system has successfully extracted and stored all the necessary information from newly created document collection. The information will be fully used in the next answer retrieval and ranking steps.

7. Answer Retrieval

After all the meaningful terms in the document collection have been successfully indexed and weighted, our QA system is ready to match some relevant documents to the received question. Certainly, as it was mentioned in the last section, the users' questions also need to be stemmed and striped off the high-frequency terms in order to match to the potential documents which have been processed in the same way. During the answer retrieval, the qualified documents which are related to the received question should be searched and sent back to the users as a group of candidate answers.

There are different strategies for searching the related documents. Using the traditional term-based document retrieval, some systems firstly distribute each term in the processed question. Each isolated term then is used as an independent token to match the same term stored in the document index. Since each term in the index has been attached with some document IDs as the term location, if a term marches with the token the system is holding with, all the documents which contain this term will be detected and sent back to the users. In other words, as long as a document shares at least one common term with the question, it will be searched as a relevant document. For example, for the question "who was the first American President", it is firstly pre-processed by the system and only "first American President" will be left. These three terms then will be used as three different tokens in the document retrieval. If a document contains the same term with the first token “President”, this document will be located and extracted from the collection. This process is repeated until all the text tokens in the received question have been matched separately. The entire matched documents will be treated as the related documents which have the potentials to answer the question.

This traditional term-based document retrieval strategy has its own advantages and disadvantages. Comparing with the expensive and sophisticated pattern learning strategy and semantic analysis introduced in the literature survey, term-based document retrieval is more straightforward. It only requires the system to check the existences of the common terms between the question and each document. Also, it works great in the case of which the ideal answer in the collection does share the common terms with the question. This strategy makes sure the system does not miss this expected answer during common term matching. However, its disadvantages are also conspicuous. This strategy is apparently too sensitive to search for the real potential document. A document which only shares one common term with the received question will definitely be searched during the retrieval. Comparing with other searched documents which share more terms with the question, the quality of this document is lower. In other words, it has a lower ability to completely answer the question. Having this kind of documents in our candidate answers wastes users time and decreases the answer quality. Also, as the side effect of our term stemming process of the document indexing, more documents are able to be matched to the questions. Even a document which shares just a common stem instead of a complete common term with the question will still be searched and treated as a potential answer. For example: the document “He answered the phone as quick as he could” will be matched with the question “what is a question answering system” due to the common stem “answer”. Clearly, this document does not answer this question at all. Therefore, our QA system which utilizes this answer retrieval strategy has also been integrated with another process to solve the sensitivity discussed here. The process will be explained in the next section.

Another disadvantage of the term-based document retrieval is the semantic information missing. The strategy is too arbitrarily to determine which documents are related to the questions. Again, only the documents which share the common terms with the questions can be retrieved. For the documents which contain the important information for answering the question but did not qualified to this comment-term requirement will be missed during the retrieval. This disadvantage predictably decreases the quality of the candidate answers as well. For example, the document “Artificial intelligence is the intelligence of machines and robots and the branch of computer science that aims to create it” is the ideal answer to the question “What is the AI”. However, since this document does not share any common terms or stems with the question, it will not be matched as the candidate answer. Therefore, to avoid the missing of the semantic information, the improvements are needed to our term-based document retrieval strategy.

Based on the Mehran and Timothy’s work [63] introduced in the related work, they have designed and implemented a simple but powerful retrieval strategy to detect the existence of the semantic connection between the question and each of the documents. Specifically, they took the advantage of the search results from the Google search engine to accomplish this task. In the search results of Google, there is a special type of information called web page snippet. Each web page snippet is shown under the web page link in the search result page. It describes the reason that the corresponding web page was matched to the search query. It contains the web page sentences' segments which share the common terms with the query. For example, if a user searches the question "what is the AI" with the Google search engine, the first web page snippet under the first matched web page is "WHAT IS ARTIFICIAL INTELLIGENCE (AI)? ... This article for the layman answers basic questions about artificial intelligence. The opinions expressed here are not". This snippet implicitly explains the reason that this web page matched with the search query since they share the common term "AI".

Furthermore, the snippets retrieved during the webpage searching contain not only the common terms matched with the question, but also the extra terms which are highly related to the question. Mehran and Timothy utilized this feature to expand the received question with the more related terms from the snippets to retrieve more related documents in the collection. Based on the example talked above, with the help of the snippet "WHAT IS ARTIFICIAL INTELLIGENCE ... This article for the layman answers basic questions about artificial intelligence… The opinions expressed here are not", the question "what is the AI" is expanded with the phrase "artificial intelligence". Therefore, the document "Artificial intelligence is the intelligence of machines and robots and the branch of computer science that aims to create it" now shares this common phrase with the expanded question. It will then be successfully retrieved by the improved term-based answer retrieval strategy.

Furthermore, Mehran and Timothy expanded not only the questions but also each document in the database with their corresponding web page snippets from the Google search engine. Each question and document is sent to the Google search engine separately as a piece of search query to acquire the snippets from the matched web pages. After receiving the snippets for a specific document or question, only top 50 terms which have the highest TF-IDF values will be saved to expand this document or question. Thus, the question and documents are all expanded with 50 related terms individually. After the expansions, more useful documents which originally did not share any common terms with the question then have a higher possibility to contain some common terms and be matched to the question.

Mehran and Timothy’s snippet search strategy is simple and powerful. Because without requiring the system to be integrated with the expansive and complicated semantic analysis, it still is able to catch the semantic connection between the documents and the question. This was the reason their work came to our attention at the first place. Since our QA system missed the semantic information by only using the traditional term-based search strategy and also the system has been receiving the text packages which contain the web page snippets from the Google search engine, it was a certain decision for us to embed Mehran and Timothy’s snippet searching strategy into our QA system.

However, there is a major limitation of this strategy. Mehran and Timothy claimed that their snippet searching strategy works for the query suggestion system. In the query suggestion system, a query collection is prepared before the system receives a new query. Each existing query in the collection will be sent to the Google search engine to get the web page snippets as its expansion materials and be expanded by those snippets. When a user submits a new query to this system, the expanded and related queries in the collection will be matched to the user’s query and shown as some suggestions to the user for a better search result. Therefore, the query suggestion system only needs to expand their query collection once during the preparation. However, as a dynamic QA system, our system initializes the document collection each time it receives a new query. In most cases, there are more than 200 documents in the collection each time. It means if our system completely follows Mehran and Timothy’s retrieval strategy, it has to send more than 200 queries to the Google search engine to expand the new received question and the new collected documents every time it answers the question. The time spent on this document preparation will be expectedly long and unaffordable to the users and our system. Also, those 200 documents will be expanded with the new related 50 terms individually. It means there are 10,000 new words adding into our document collection. Our QA system will have to spend more time on document retrieval and ranking in the next steps.

Therefore, we have modified and improved Mehran and Timothy’s searching strategy in order to embed it into the dynamic QA systems like ours. We keep the step which is expanding the users’ questions by the web page snippets. Since our system has been designed to send the received questions to the Google search engine as the search queries, those questions then will be expanded by the corresponding web page snippets extracted from the text packages the Google search engine sends back.

Furthermore, there are two main reasons we do not further expand the documents in our collection after the question expansion. First of all, as it was explained above, our QA system cannot afford to send more than 200 documents to the Google search engine every time in order to answer a particular question. The time complexity is too big to be accepted. Secondly, the query suggestion system which Mehran and Timothy’s strategy works well with collects the queries from different resources and builds up its own query collection before receiving any questions from the users. Thus, the queries the query suggestion system collects originally have no connections between each other or with the future users' own queries. On the contrary, the documents stored in our QA system are collected based on each question the system receives. Our QA system finds those documents only because the Google search engine searched them from the Internet based on the specific question. These documents were the paragraphs on the searched web pages. Thus, the documents and the question are implicitly connected in our QA system at the beginning. If our system also sends those documents to the Google search engine as the search queries to expand them with their web pages snippets. Most of the document including some irrelevant ones will be matched to the users' question at the end due to their new expansions. For example: in Figure 4 the document d_5 which was a paragraph from the web page w_1 is actually not related to the question q. The web page $w_1$was searched by the Google search engine based on the question q. Therefore, the question q will get expanded as the strategy by the snippet s_1 from the web page w_1. If we also expand the document d_5 by sending it back to the Google search engine to search for the related web pages and the web page snippets, the web page w_1 to which the document $d_5$belongs will definitely be searched again. Then the document d_5 will get expanded by the snippet s_2 from the same web page w_1 . Thus, there is a high possibility that the snippets s_1 and s_2 share some common terms since they are all from web page w_1. And the expanded irrelevant document d_5 will possibly be matched to the expanded question q by the term-based document retrieval strategy.

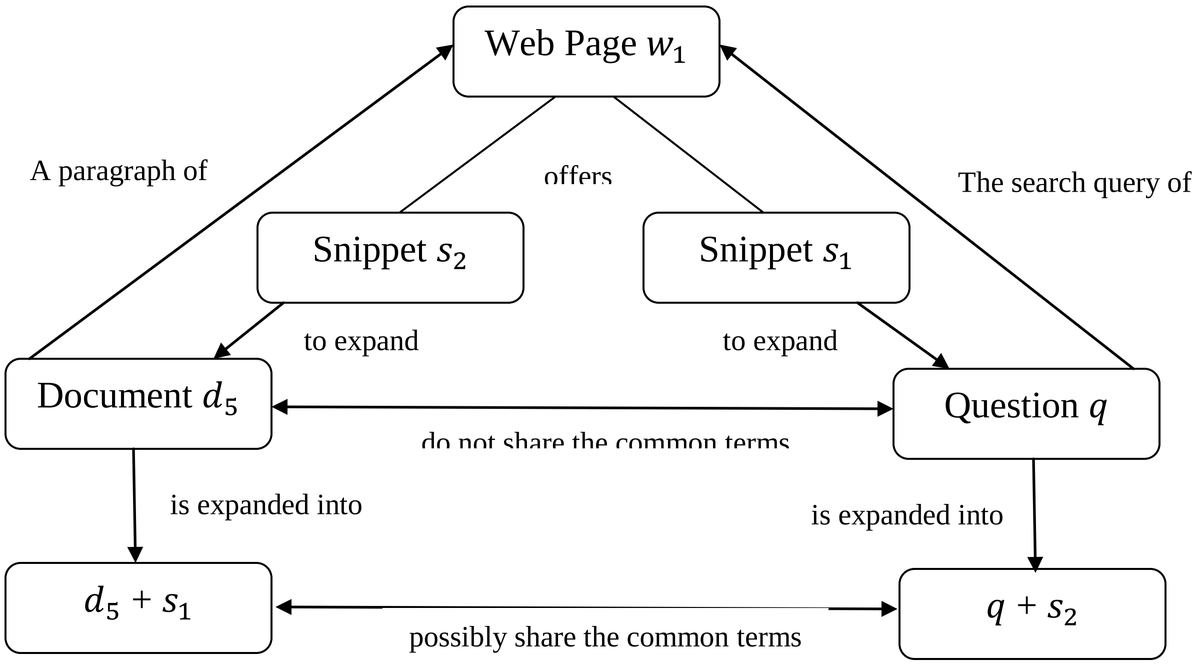


Figure 4

Based on those two reasons, our QA system only expand the users' questions by the web page snippets extracted from the text package the Google search engine sends back. No documents in the document collection are expanded. After the question expansion, the traditional term-based answer retrieval strategy will be executed as being explained at the first section. In the evaluation section, the remarkable improvement made by our modified snippet search strategy will be shown to prove its ability.

8. Candidate Answer Ranking

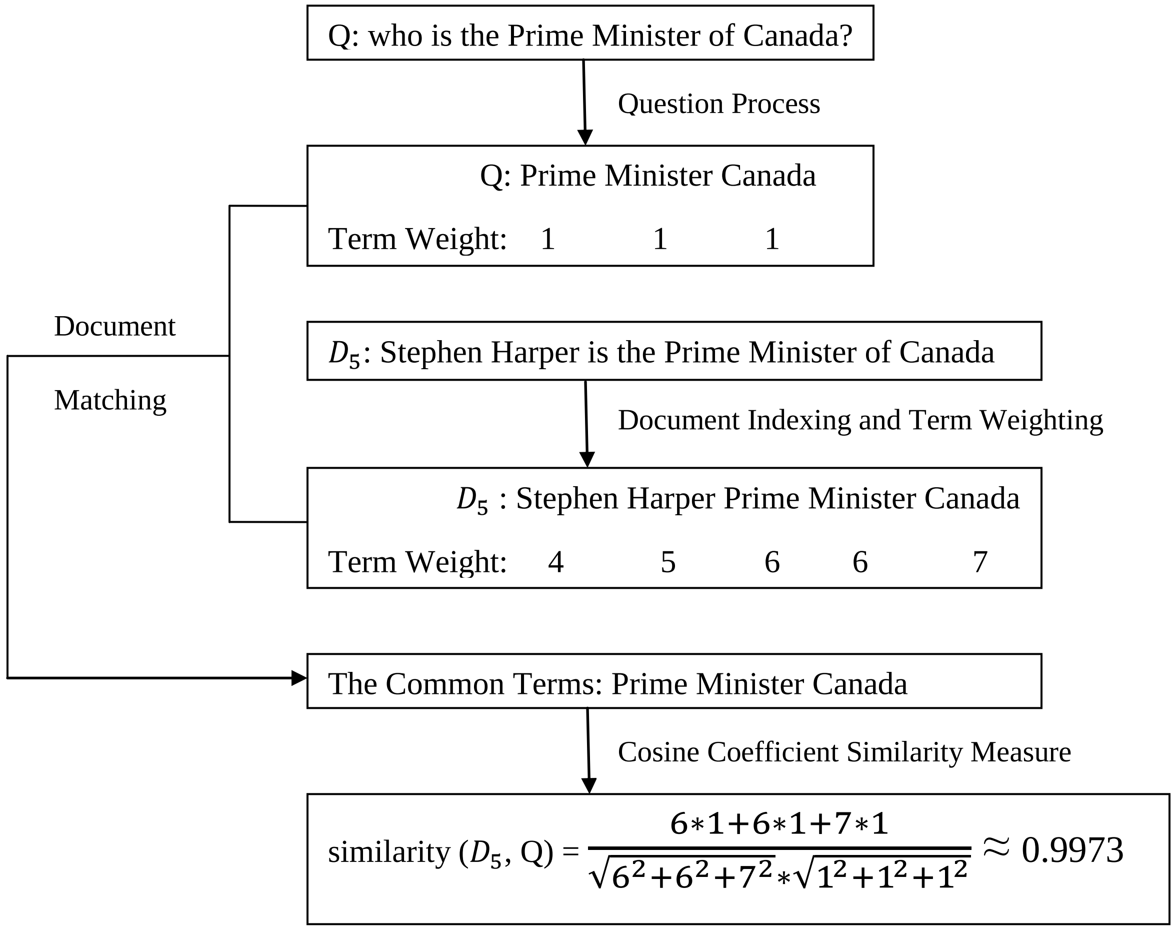
Our QA system also ranks the retrieved the documents which share one or more common terms with the expanded question. The ranking process is based on the similarity between each retrieved document and the question. After the ranking, each retrieved document has been evaluated and assigned with a numeric value indicating how similar and related it is to the question. The ranking result is presented back to the users in descending order of the retrieved documents' similarity values. Thus, a document which is more similar to a user's question gets a higher rank. The documents with the higher ranks are expected to have stronger abilities to answer the question and will be viewed earlier by the users.

There are two reasons of using ranking strategy after the document retrieval. First of all, the ranking process helps the users find their answers with fewer efforts. In most cases, users do not need to go though the entire retrieved documents. They will find the expected answers among the top 10 to 20 retrieved documents. But without the ranking process, our QA system will only send the searched documents to the users in a random order. The worst case then will be the users find the correct answer at the last document they are viewing with. Moreover, the ranking process also solves the sensitivity of the term-based document retrieval strategy. As it was explained in the last section, as long as a document shares at least one common term with the question, it will be considered as a related document with the question and sent back to the users. Therefore, the users of our QA system receive the searched documents more than they need. It is not efficient to them to go through all of the retrieved documents looking for the best answer. It is neither the satisfied result our system can accept. Therefore, as it was talked before, the system has been integrated with a list of stop words to prevent the document search strategy from matching too many irrelevant documents to the questions. The high-frequency terms on the list such like "what", "here", "and", etc. are not treated as the useful common terms and will not be able to largely raise the number of retrieved documents. Here, the ranking strategy is another important way to narrow down the number of retrieved documents the users need to go through and improve the sensitivity of the term-based document retrieval eventually. The irrelevant ones and less related ones among all the retrieved documents will be ranked at the lower positions in the list. Before the users go through them, the users have probably found the answer they need.

The Cosine Coefficient Similarity Measure (CCSM) [68] is one of the traditional measures to evaluate the similarity between two text documents. In our case, the questions from the users are treated as the short documents to compare the similarities with the searched documents from our collection. The CCSM utilizes the TF-IDF term weighting value to determine how similar two documents are. Here is the formula of the CCSM:

$$similarity\ (j, q) = \frac{\sum_{i=1}^{n} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{n} {w_{ij}}^2} * \sqrt{\sum_{i=1}^{n} {w_{iq}}^2}}$$

In the formula, the *q* represents the question; the *j* is the index of the searched document. Moreover, the *i* indicates the index of the common term which is shared by the document *j* and the question *q*. There are *n* common terms between the document *j* and the question *q*. Therefore, the w_{ij} is the weighting value of the term *i* in the document *j*; the w_{iq} is the weight of the term *i* in the question *q*. In our QA system, the TF-IDF weight of each term in the different documents has been evaluated and stored during the document indexing. Each weight of the term in the question has been initialized with the weight value 1 in order to treat them equally during the initial candidate answer ranking. The users later will be asked about their personal opinions about each term weight in their questions during the question refinement step. The new weights will be attached to the questions' terms after the users finish that step. And a new similarity measurements and ranking result should be made based on the new weights the question terms got. This extra step will be fully explained in the next section. Here, in the initial searched document similarity measurement, the prepared term weights of the common terms will be used to calculate the similarity between each document and the question. Here is an example in Figure 5 to better explain this calculation. In this example, the similarity value between the question Q and the document D_5 is measured by using the CCSM. The similarity is about 0.9973. Since the similarity value range of the CCSM is from 0 to 1, the document D_5 then is considered as highly similar to the question Q.


Q: who is the Prime Minister of Canada?
Question Process
Q: Prime Minister Canada
Term Weight: 1 1 1
Document Matching
D_5: Stephen Harper is the Prime Minister of Canada
Document Indexing and Term Weighting
D_5 : Stephen Harper Prime Minister Canada
Term Weight: 4 5 6 6 7
The Common Terms: Prime Minister Canada
Cosine Coefficient Similarity Measure

$$\text{similarity}(D_5, \text{Q}) = \frac{6*1+6*1+7*1}{\sqrt{6^2+6^2+7^2}*\sqrt{1^2+1^2+1^2}} \approx 0.9973$$

Figure 5

That is how Our QA system utilizes the original CCSM to measure the similarities between the expanded question and the retrieved documents and rank those documents based on their similarity values. However, our system is not just dealing with some general text documents in its collection, it actually retrieves and ranks some carefully collected and prepared documents. Those documents used to be the paragraphs in the related web pages searched by the Google search engine with the users' question. Therefore, we have also modified the original CCSM to adjust it to work better with our special documents for not missing their specialties.

First of all, the quality of the web page each document was downloaded from is treated as an important factor in our Modified Cosine Coefficient Similarity Measure (MCCSM). Since our system receives a list of ranked links from the Google search engine each time, it treats the rank information as the indicator of the corresponding web page quality. A web page link gets a higher rank means this web page has more related text content to the search query which is the users' question. Therefore, the documents in our collection are firstly grouped by their original web pages' ranks. It is because the text content on each web page is distributed by our system based on its natural paragraph structure. Thus, if our system receives the top 10 web page links from the Google search engine, then there will be 10 groups of paragraphs saved in to our document collections. Those 10 groups of documents have different qualities based on the list of ranked links. Our system stores the corresponding the web page's rank information into a document's name during the document preparation. While a retrieved document is evaluated for the similarity with the question, this quality information will be exacted from its file name and used into the similarity measurement. The formula below describes the MCCSM which has been adjusted by the information of the web page ranks.

$$similarity\ (j, q) = \frac{\sum_{i=1}^{n} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}{}^2} * \sqrt{\sum_{i=1}^{n} w_{iq}{}^2}} * \frac{1}{\sqrt{r_j}}$$

In the formula, besides the variables introduced above, the new variable r_q is the rank of the web page the document *j* was downloaded from. Because the range of original CCSM is from 0 to 1, the rank information is first normalized as $\frac{1}{\sqrt{r_j}}$. It is also because our QA system treats this factor as just one of the references to calculate the document similarity. It is supposed to be working with other factors to determine the similarity result. For example, if the document *j* is from the web page which is the second one on the ranked list, then the original similarity between the document *j* and the question *q* will be multiplied with the value of 0.87. However, if the web page is the first one on the list, then the similarity of the document *j* will be multiplied by the value of 1. Thus, the rank is higher; the similarity of the document gets bigger. This example is also explained in Figure 6.

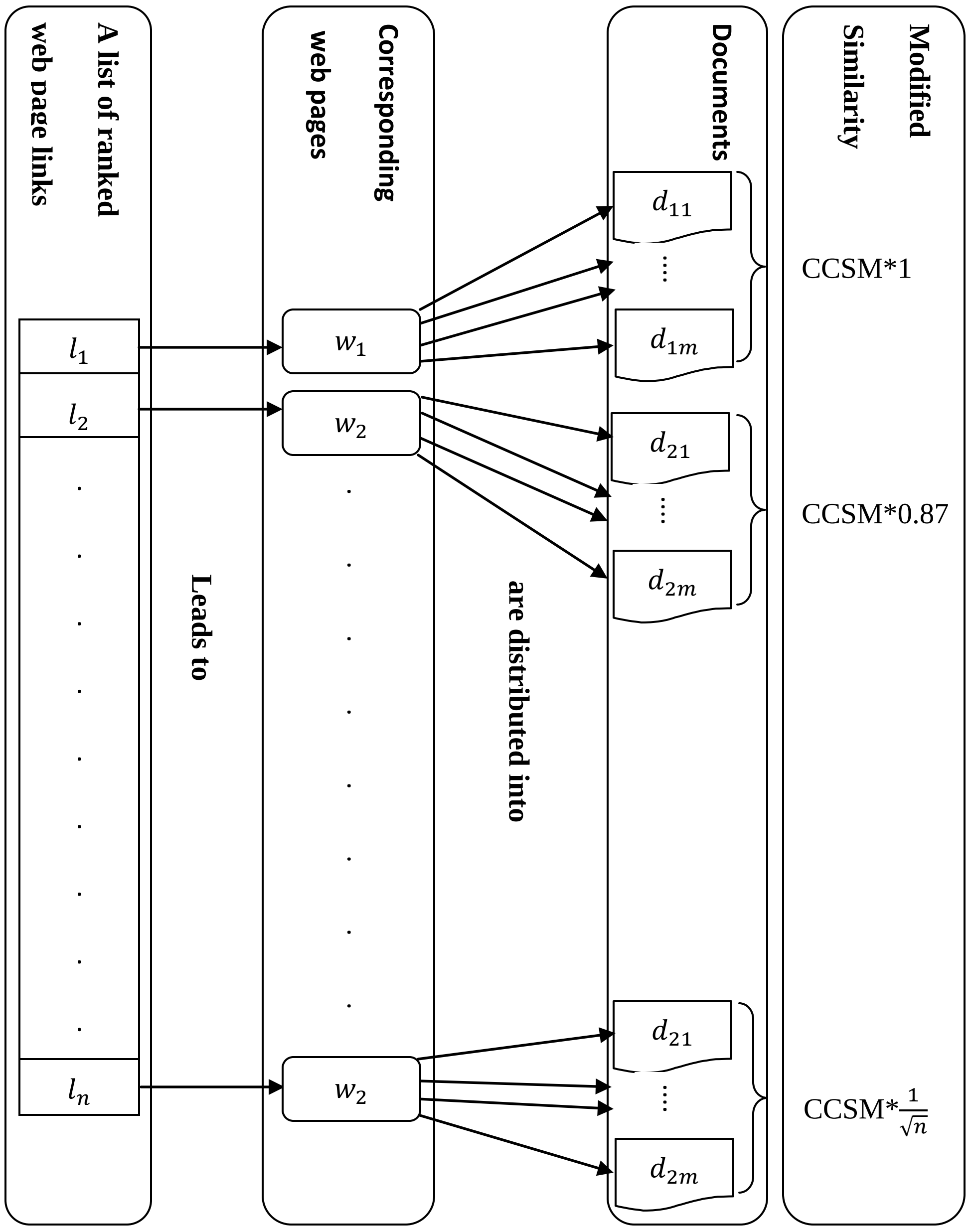


Figure 6

Moreover, since the documents were the paragraphs on the web pages, the length of each retrieved document is also being considered into our similarity measurement. Based on our observation, a longer paragraph on a web page mostly contains more valuable information. Again, like the rank information of the searched web pages, the length of a specific retrieved document is only one of the factors which are affecting the similarity result. When we were embedding this factor into our MCCSM formula, we carefully used the normalization to allow it only genteelly affect our result. The formula is shown below.

$$similarity\ (j, q) = \frac{\sum_{i=1}^{n} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} * \sqrt{\sum_{i=1}^{n} w_{iq}^2}} * \frac{1}{\sqrt{r_j}} * (1 - \frac{1}{l_j})$$

The l_j in the formula is the length of the document *j*. It denotes the number of identical terms in the document *j*. The high-frequency terms have been eliminated from the document *j*. And the repeating terms in the document *j* are counted as one term in this formula.

Thus, the formula below is the ultimate MCCSM our QA system uses to calculate the similarity between each retrieved document and the question. It has been carefully integrated with the web page rank information and the document length based on the special features of our document collection. The improvement of these two modifications will be presented in the evaluation section.

$$similarity\ (j, q) = \frac{(l_j - 1) * (\sum_{i=1}^{n} w_{ij} * w_{iq})}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} * \sqrt{\sum_{i=1}^{n} w_{iq}^2} * \sqrt{r_j} * l_j}$$

9. Query Refinement

In our QA system, after the users receive and go through a list of ranked candidate answers, they will be asked for their opinions about the quality of those answers by the question “Did those documents answer your question”. If the answer is yes, that means the users have been satisfied with the search results and they have found the useful answers in an acceptable time. If the answer is no, it means either our QA system have not successfully retrieved the expected answer for the users or the users did not find it among as many candidate answers as they can go through. The users then will be offered an opportunity to refine the query by giving more explicit and useful search clues to our system to change the components and the ranks of the next search result. The next retrieved candidate answers are expected to have more correct answers with the higher ranks.

There are three steps in the query refinement for the users to follow. The first one is asking the users to reedit their text questions. The users can add and eliminate some necessary and useless terms respectively in their questions. Different from those popular web page search engines' advance search pages, the users in our QA system have the previews search results on the same page with the refinement as their references of making changes to their queries. We believe this special improvement helps our users better understand the influence of each term they typed in their questions last time and make better questions this time. For example, if a user who is seeking for the information that the population of London in Ontario asked the question "what is the population of London" at the first time, the most of the candidate answers this user got for the first time is the population of the city London in United Kingdom. By using this search result as the reference of the query refinement, this user will learn to specify the city into the province of Ontario. And he or she will probably get the expected information next time. However, this step is not mandatory to the users to follow. They can skip this one and make the changes in the next two steps. Therefore, if the query has not been revised at all, our QA system will not send it to the Google search engine again as a new search query. Because the new Google search result would still be the same if our system did. And the new prepared document collection would be the same with the last collection. Therefore, the previews document collection and the index will be kept in our system to avoid downloading the same related web pages again and save the answer retrieval time eventually. If the users did make some changes to their queries, then the new queries will be sent to the Google search engine to acquire the new related web pages as the new knowledge source. Then both the previews document collection and the index will be replaced by the new ones. At last, the new candidate answers will be searched from the new document collection based on the new question. Therefore, the query reediting completely affects the entire search result including the members of the candidate answers and their ranks.

The second step in the query refinement is letting the users evaluate each term in their questions and assign a numeric value from 1 to 10 to it. The users can attach the different values to the different terms in the questions based on their opinions of how important those terms are in their queries respectively. In other words, the new term values of the questions' terms will be treated as the new term weights in the next candidate answer ranking process. As it was mentioned before, our QA system initializes the weight of each term in the question with the value of 1 in order to treat them equally in the first candidate answer ranking with the MCCSM. In this query refinement, the users can emphasized some important terms by attaching some bigger term weights to those terms. Our system will use these new weights to list the documents which are more related to those emphasized terms with the higher ranks. For example, the question "who is the first

President of America to visit Europe" retrieves also some documents which introduce the Presidents in Europe. If the user who asked this question attached a bigger weight to the term "America" than "Europe", the documents which contain the information about the President of America will be ranked higher than the documents which introduce the Presidents in Europe. The user then can find some useful retrieved documents with less time spent. Thus, the question term weighting does not change the members of the candidate answers but the ranks of them.

The last step of the query refinement is binding a Boolean query with the term-based query. As the disadvantage of the traditional term-based document search strategy our QA system uses, the strategy does not offer the function as the Boolean operator "NOT" does. It only searches the documents which the users are willing to read. It does not filter the documents which the users do not expect to see specifically. That is why we can utilize the function of the Boolean operator "NOT" to help the users indicate one or more terms which are not expected to be seen in the retrieval result. In other words, as long as a document which contains one of the terms mentioned in the Boolean query, this document should not be retrieved even if it also shares some common terms with the question. The users in this step can build up their own Boolean queries with the unwelcomed terms. For example, a user asked the question "how to cook some sweet corns" in our QA system. Our system then sends this question to the Google search engine to search for some relevant web pages. During the Google search process, some web pages which contain the recipe of cooking the salty corns are considered as the related web pages to the query and will be shown in the Google search result. Our QA system then will download those web pages and save their text contents into our document collection. Thus, since some documents which came from the salty corn web pages share the common terms “cook” and “corns” with the question, they will definitely be retrieved by the system and presented to the user. However, they are clearly not the recipes the user is looking for and their existences in the candidate answers may disturb the user’s answer searching. Therefore, the user can use the Boolean query “NOT” to restrict those documents showing in the candidate answers. For doing it, the user can simply add the term “salty” in the Boolean “NOT” query. In the next answer retrieve process, as long as a document which contains the term “salty”, it will not be retrieved by our QA system or shown back to the user. Thus, same with the question reediting but different from the question term weighting, the Boolean query changes the members of the candidate answers.

Again, same with the first step in this query refinement, the second and third step are also not mandatory to the users to finish. Those three steps are all independent and affect the future search result separately. This query refinement will be executed repeatedly until the users find their expected answers.

10. Implementation

In order to test and make use of the MCCSM algorithm and the modified and improved term-based document search strategy, we have successfully implemented them together as a complete and functioning QA system. This QA system is dynamically connected with the Google search engine to access the abundant and up-to-date online information. Users can ask their questions in natural English on the unlimited topics to our system. Based their questions, our system is able to answer them with numbers of related and ranked documents as the candidate answers. Thus, the users can find the needed and correct information among those retrieved documents with less time spent. The QA system then is able to improve the quality of the online information retrieval eventually.

During the implementation, we used Java with the NetBeans IDE 7.0.1 as our programming language and the coding environment. NetBeans offered us a visualized coding environment so that we could program and design the User Interface at the same time with a more straightforward view. The class-based and object-oriented features of Java programming language offered us an ideal programming structure to use to implement our algorithms and strategies. We were able to code the different steps of the process as the different classes in the separate objects. Thus, it is easier and clearer to monitor and test their functions and performance individually. It will be also easier to control the affections of the future modifications. Furthermore, the various useful code libraries of Java were the great helps to us during the coding. We could utilize the different libraries' existing functions into our program to be more efficient.

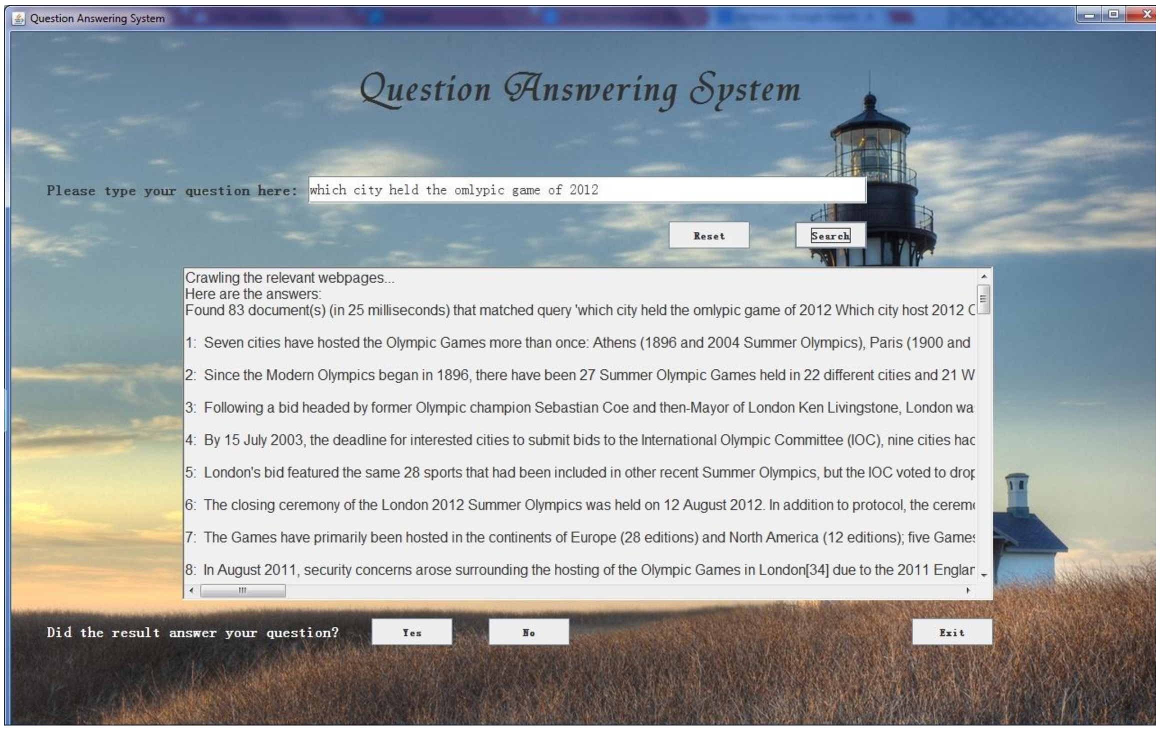
Lucene [69] is the main code library we used in our program. It is an information retrieval library that helps different applications with their local document searching functions. There were two reasons we chose Lucene to be one of our external code libraries. Firstly, it is easy to be embedded into most of the projects. The coding package of Lucene is only around 11 MB large. As long as it has been imported into a program, the basic searching functions can be implemented with just several lines. Also, Lucene offers the profession users to overwrite some of its main classes including the document ranking class. Researchers like us are able to implement their own document retrieval and ranking algorithms by overwriting them on the Lucene's corresponding classes.

Therefore, in our QA system, we have imported Lucene into our document indexing process and overwritten it with our own strategy. We use Lucene to index the text documents once they are downloaded into our collection. Lucene is able to locate each identical text term in each document and store their locations into the index file which is saved as a local file for the future process. Instead of using the standard analyzer of Lucene to detect the term while indexing, we imported a new analyzer to stem each term and eliminate the high-frequency words as the same time. Thus, the terms indexed by our system have been stemmed as we explained in the document indexing section. Meanwhile, there are no high-frequency terms such as "in", "I", "it", etc. existing in the index. Furthermore, we utilized a special feature of Lucene to save the rank of the web page each document was extracted from. The web pages' ranks, as we explained in the document preparation section, have been saved into the documents' file names. During the document indexing, this information is extracted from the documents' names again and saved into the index file indicating how potential those documents may be individually. It is prepared for the future candidate answer ranking process.

Our system also utilized Lucene in the document searching and ranking process. The received question is saved as the search query. Apparently, the search query should be processed firstly with the stemming strategy and the list of high-frequency words before the document matching. It is because the indexed terms stored in the documents have been stemmed in the same way. The search query can only match them if it is stemmed as well. Moreover, we delete the high-frequency words from the query for retrieving less irrelevant documents. Our system then goes through the index files to match the common stems between a document and the processed search query. After all the documents which share as least one common term with the search query have been retrieved by our system, they will be scored and ranked based on how similar they are to the query. Lucene has its own similarity measure algorithm which was originally developed from the CCSM. However, it also offers an option to the professional users to overwrite their different similarity measure algorithms on the existing class. Therefore, our system has overwritten the default similarity measurement with the MCCSM to rank the candidate answers based on our own algorithm. Specifically, as it was explained in section of the candidate answer ranking, there are four variables in our MCCSM: TF, IDF, the web page rank and the document length. We use Lucene to calculate the values of TF and IDF. The web page rank and the document length of each document were prepared additionally during the document indexing. Thus, our QA system can use the values of those four variables in the MCCSM to calculate the documents' similarities and rank those documents for the users to view.

Our QA system does not have to set up a formal database based on the system demands: all the local files created during the process need to be initialized every time our system receives a new query; usually only 350 documents are processed by the system at a time. The user's inputs, the intermediate products and the final outputs are saved separately in the different text files. In the document collection, the extracted documents from the web pages are named with the five identical digits. The document index information is saved in the index files. There are also a few more text documents created by our system to save the users' questions, the web page snippets and the information of the query refinement. Their purposes and usages will be explained below.

As it was mentioned above, we used the NetBeans IDE 7.0.1 as a Graphic User Interface (GUI) builder to design the UI of our QA system. The first original search page is shown in Figure 7. It is the first page our system shows to the users for interacting with them. They can type in their text questions in the search box. The unnecessary symbols such as "!", "?" and ".", etc. in the questions will be eliminated before they are sent to the Google search engine. Our system also converts the users' questions into the formal search queries which the Google search engine reads only. The whole searching process is started once the users click on the search button. After our system receives the Google's search result back, the complete documents preparation, document retrieval and candidate answer ranking are going to be accomplished one by one. The ranked candidate answers then will be shown in the text box for the users to view. Moreover, the question "did the result answer your question" shows up along with the candidate answers, the users can answer it with the yes or no bottoms. If they click on yes bottom, a small new window will pop out welcoming them to go back and search more questions in our system. The users can then use the reset button to clean the previews question and candidate answers. If they choose no, the advanced search page for the query refinement will be shown to them. The current original search window will be closed automatically. Last, the users can end their search process by clicking on the exit bottom.



Please type your question here: which city held the omlypic game of 2012

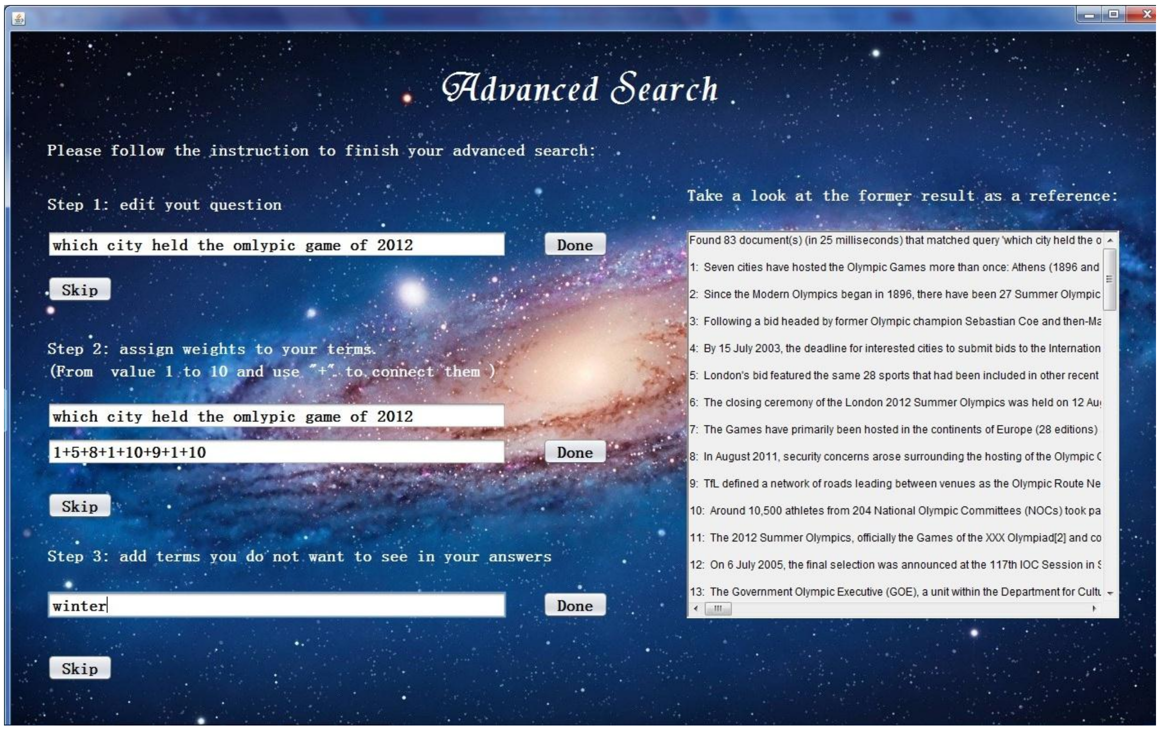
```
Crawling the relevant webpages...
Here are the answers:
Found 83 document(s) (in 25 milliseconds) that matched query 'which city held the omlypic game of 2012 Which city host 2012 C

1:  Seven cities have hosted the Olympic Games more than once: Athens (1896 and 2004 Summer Olympics), Paris (1900 and

2:  Since the Modern Olympics began in 1896, there have been 27 Summer Olympic Games held in 22 different cities and 21 W

3:  Following a bid headed by former Olympic champion Sebastian Coe and then-Mayor of London Ken Livingstone, London wa

4:  By 15 July 2003, the deadline for interested cities to submit bids to the International Olympic Committee (IOC), nine cities hac

5:  London's bid featured the same 28 sports that had been included in other recent Summer Olympics, but the IOC voted to drop

6:  The closing ceremony of the London 2012 Summer Olympics was held on 12 August 2012. In addition to protocol, the ceremo

7:  The Games have primarily been hosted in the continents of Europe (28 editions) and North America (12 editions); five Games

8:  In August 2011, security concerns arose surrounding the hosting of the Olympic Games in London[34] due to the 2011 Englar
```

Did the result answer your question?

Figure 7

While the users are asking and searching their questions in the original search page, there are different documents created to serve the different functions. The query document saves the current original question from a user. This document will be read again in the advanced search page to show the user’s original question and let him or her revise it. A document called Google feedback will be created for saving the text package which includes the relevant web page links from the Google search engine. After the system detects and extracts the needed information from this document: a list of ranked web page links and the web page snippets, those two types of information will be saved into two documents for the next process. Furthermore, our QA system reads and presents the candidate answers which are retrieved based on a user’s question and saved in the result document.

In the advanced search page shown in Figure 8, the users who were not satisfied by the previews candidate answers will have a simple and straight forward instruction to help them modify their questions on this page. They will be asked to reedit their question first. Their original question has been shown in the text box once this page is popped out so that the users do not have to remember what exactly they asked before. After they finish editing the terms in their queries, they can click on the done bottom to execute the next step. The second step of query refinement is assigning different numeric weights to each term of the users' newly revised questions. By following the instruction, the users will be informed that for typing the term weights in the text box each term weight should be in the range of 1 to 10 and connected by the "+" operator. Their inputs of this step will be checked by our system in case some terms have not received the proper wrights. The users will be notified if there are some illegal inputs. They can only see the done bottom when their inputs have been checked and successfully qualified. The last step of this page is binding an extra Boolean "NOT" query with the text question. As it was explained in the section of query refinement, the users can type in the term they do not expect to see in the future candidate answers. While the users are refining their questions on this page, they also have the previews search result shown on the right side of the window. By having it as a reference, the users will better understand the power of their words during the answer retrieval and carefully submit a more robust question next time. Those three steps of the refinement are all optional. The users can simply skip some of them by clicking the skip bottoms under each text box.



Advanced Search

Please follow the instruction to finish your advanced search:

Step 1: edit yout question

which city held the omlypic game of 2012

Step 2: assign weights to your terms.
(From value 1 to 10 and use "+" to connect them)

which city held the omlypic game of 2012

1+5+8+1+10+9+1+10

Step 3: add terms you do not want to see in your answers

winter

Take a look at the former result as a reference:

```
Found 83 document(s) (in 25 milliseconds) that matched query 'which city held the o
1:  Seven cities have hosted the Olympic Games more than once: Athens (1896 and
2:  Since the Modern Olympics began in 1896, there have been 27 Summer Olympic
3:  Following a bid headed by former Olympic champion Sebastian Coe and then-Ma
4:  By 15 July 2003, the deadline for interested cities to submit bids to the Internation
5:  London's bid featured the same 28 sports that had been included in other recent
6:  The closing ceremony of the London 2012 Summer Olympics was held on 12 Au
7:  The Games have primarily been hosted in the continents of Europe (28 editions)
8:  In August 2011, security concerns arose surrounding the hosting of the Olympic C
9:  TfL defined a network of roads leading between venues as the Olympic Route Ne
10:  Around 10,500 athletes from 204 National Olympic Committees (NOCs) took pa
11:  The 2012 Summer Olympics, officially the Games of the XXX Olympiad[2] and co
12:  On 6 July 2005, the final selection was announced at the 117th IOC Session in S
13:  The Government Olympic Executive (GOE), a unit within the Department for Cultu
```

Figure 8

On this advanced search page, at most three new documents are created during the users' refinement. The revised new query will be still saved into the same query document to replace the previews one. This document will be read again when the new query is ready to be search. If a user chooses to skip this step, a new document indicating that there is no change for the last query will be created and saved. Our QA system then will be informed by this document that no new query needs to be sent to the Google search engine again. Then the answer retrieval will be executed based on the same document collection extracted last time. Moreover, the new term weights created in the second step will be saved into the document called query weights for the usage of the next document retrieval. Similarly, the terms which are going to compose the Boolean "NOT" query are stored in the NOT document.

After the users finished all three steps on the refinement page, the newly initialized original search page will replace this page and the new or old question will be shown in the search box. The users only need to click to the search bottom to execute a new answer search process. All the new made modifications, query term weights and the Boolean "NOT" query will be involved during the next retrieval for the better candidate answers. The users can keep refining their questions by clicking the no bottom to call the advanced search page several times. We have designed the exit bottom and the window close bottom to help them halt the searching circulation.

11. Evaluations

Since we have modified both the traditional term-based document search strategy by expanding the search queries and the CCSM algorithm to calculate the similarity of the retrieved documents from the web pages, we needed to design two evaluation schemes to test those two modified algorithms separately. Instead of having subjective surveys collecting the users' experiences and opinions of our system, we preferred to evaluate them with some scalable and objective evaluations. Only the figures can prove the improvements and the performances of those two modified algorithms.

11.1 Evaluation on Document Retrieval Strategy

To evaluate our modified and improved document retrieval strategy which expands the received questions in our QA system, there are two factors need to be considered. First of all, one of them is how many relevant documents our system is able to retrieve from the document collection based on a question. The purpose of designing the search query expansion which is about adding the web page snippets to the query is helping our system retrieve more semantically related documents from the collection. Therefore, we are expecting our system is able to retrieve more relevant documents with the modified document retrieval strategy than using the traditional strategy. Another factor to consider the improvement is the number of irrelevant documents our system searches back. Since the raise of the number of the retrieved documents is not avoidable in our strategy, we need to calculate how many irrelevant documents are also sent back to the users which will disturb them searching their ideal answers.

Therefore, we utilized two evaluation formulas to scale the performance and the improvement of our new document search strategy. The first formula is called recall [68]. It is shown below.

$$\text{recall} = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ relevant\ documents\ in\ the\ document\ collection}$$

Thus, in our case, recall is the proportion of the relevant documents retrieved by our QA system among all the relevant documents in the local collection based on a specific question. It scales our QA system’s searching ability. As a QA system, our system is supposed to retrieve the related documents to the question as many as possible to fully answer the question. As it has been seen in the formula, the higher the value of recall is; the stronger searching ability our system performs.

The second formula involved in our evaluation is called precision [68]. The formula is shown below.

$$\text{precision} = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ documents\ retrieved}$$

Thus, the precision indicates the ratio of retrieved relevant documents to the entire retrieved documents of our system. It scales the how precise our QA system is able to search for the useful documents based on a question. The value of the precision is higher means the system can retrieve more relevant documents and less irrelevant ones. Since there is no document retrieval strategy can perfectly retrieve the entire relevant documents without also retrieving some irrelevant ones at the same time, an information system which tries to retrieve more documents to raise the proportion of retrieved relevant documents gets a higher recall value but lower precision. It means the more relevant documents it retrieves, the more irrelevant documents it gets as well. Therefore, the precision evaluation value shares an inverse ratio growth with the recall value. We used both of the recall and precision formulas to evaluate our new document search strategy to have a more general view of its performance.

In order to evaluate our document retrieval strategy with the recall and precision introduced above, we need to go through the retrieved candidate answers each time we are testing our system with a question. However, our QA system builds up a temporary document collection ever time for each testing question. That means we have to go through each document in the collection to determine whether it is related to a specific question and supposed to be retrieved. If we test our system with 50 questions, we will have to evaluate 50 document collections with around 350 documents in each of them. Therefore, we needed a group of carefully collected documents as our local document collection and some designed questions based on those documents. Each question's relevant documents in the collection should be already known before the answer retrieval. Then we are able to scale how many relevant documents our system retrieved by comparing with a standard answer list of each question to calculate the value of recall and precision. Furthermore, building up such a document collection with some answered questions is difficult and expensive. It involves a number of professional researchers to completely understand and correctly evaluate each document in the collection several times in order to determine whether it is related to each designed question.

Therefore, we were using a document collection and some questions from SMART[1]. SMART have designed six groups of text articles and corresponding questions for the researchers to test their document retrieval systems. It also offers the relations between each document and each question in each group. The relations indicate the relevant documents to the different questions. We imported one group of articles as our local document collection and the corresponding questions as the search queries. In this document collection, there are 82 documents with total 2402 terms, SMART have designed 35 testing questions and their document relations in this collection. SMART defines a relevant document to a specific questions not based on the number of common terms they share together. A document is only related to a question if it is semantically relevant to the question. We used our QA system to answer each of the questions based on this document collection and compared the retrieved documents with its document relation. For comparing the different results between our new document search strategy and the traditional strategy, we answered each question twice. At the first time we simply used the testing questions as the search queries. It was the way of the traditional document retrieval strategy. At the second time we sent each question to the Google search engine to receive the snippets of its related web pages. Then we expanded the

[1] ftp://ftp.cs.cornell.edu/pub/smart/

question with the received snippets and used the expanded question as the search query to retrieve the documents. Each time, we recorded the total number of documents the system retrieved and the number of relevant documents retrieved based on the question document relations SMART offered us. Therefore, we were able to calculate the values recall and precision after answering each question with the two different strategies. Here are the evaluation results we acquired.

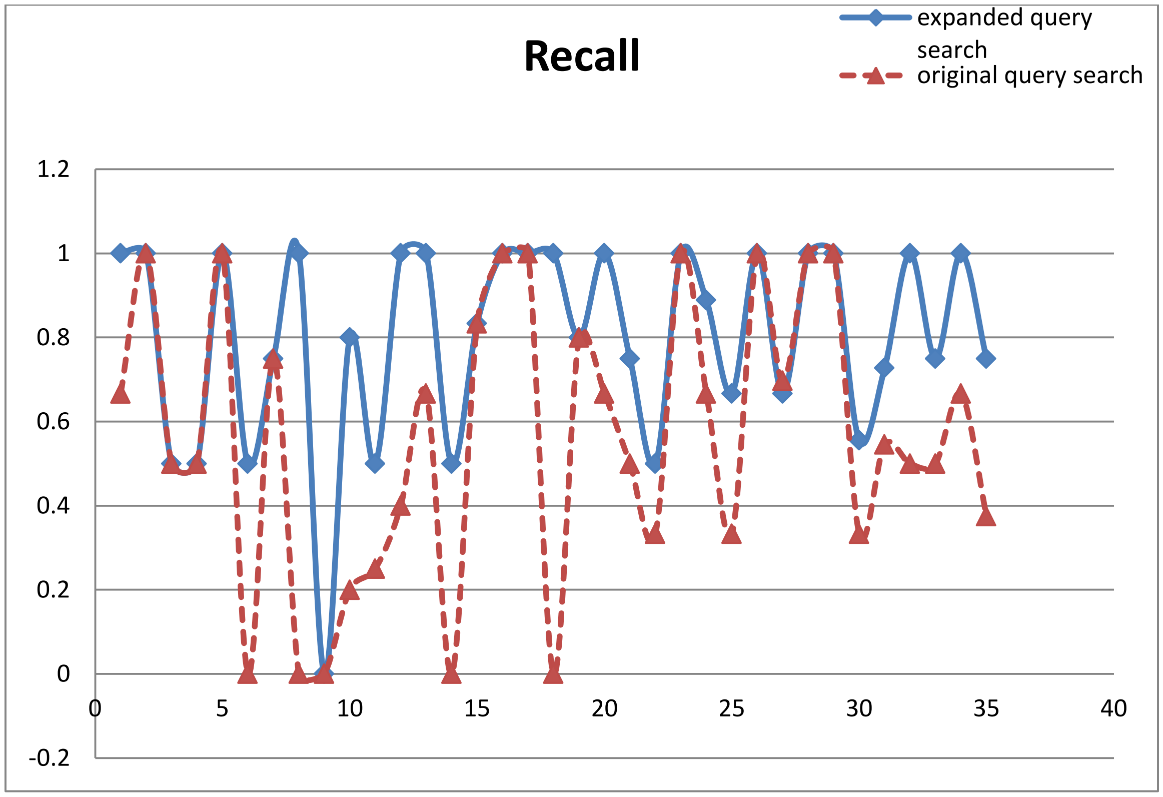


Figure 9

Figure 9 shown above is comprised by two groups of recall values. The dotted line with the triangle markers represents the 35 recall values produced by the traditional document retrieval strategy which only searches the documents with the original queries. The real line with the diamond markers denotes another 35 recalls which were calculated for our new retrieval strategy which uses the expanded queries. The X axis is the recall range from 0 to 1. The Y axis represents the index of each question our system answered. As it has been seen in Figure 9, to each question, the recall of the new retrieval strategy was equal or bigger than the recall of the traditional strategy. When the new strategy's recall was higher than the traditional one's, it means our system retrieved more relevant documents by using the new strategy than the traditional strategy. When the two recalls were the same, it means expanding the question did not help our system retrieve more relevant documents. In other words, expanding the question was either not necessary since the traditional strategy had retrieved all the relevant documents based on a testing question or not helpful to our system to retrieve more relevant documents in the collection when the traditional strategy did not retrieved them all. As a matter of fact, over the 35 designed questions, the average recall of the new strategy is approximate 0.80. But the average recall of the traditional strategy is around 0.56. Thus, our new document retrieval strategy helped the system retrieve 1.4 times more relevant documents from the collection than the traditional strategy did.

We have also calculated the 35 pairs of precision values by answering each question with those two retrieval strategies. In Figure 10, the real line with the diamond markers represents the recalls of new retrieval strategy and the dotted line with the triangle markers denotes the recalls of the traditional strategy. The X axis denotes the value of each precision and the Y axis indicates the index of each question. Each time, the new strategy's precision was really close but still higher than the precision of the traditional strategy. For answering all 35 questions, the average precision of the new strategy was about 0.050 and the traditional one's was around 0.048. It means our new document retrieval strategy helped our system retrieve less irrelevant documents than the original strategy did.

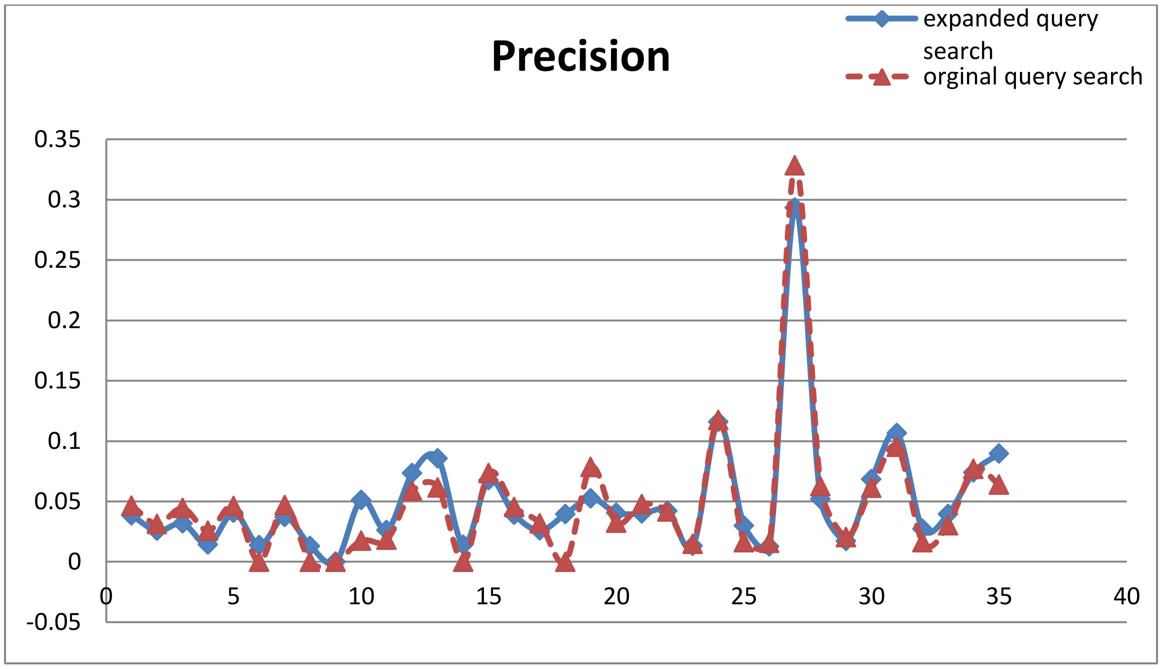


Figure 10

Comparing with the two averages of recalls and precisions, we found some unexpected results. Since the new retrieval strategy offered our system a 1.4 times higher average recall than the traditional strategy did, it means the new strategy helped our system retrieve more relevant documents. Based on the theory, the new strategy was also supposed to bring the system the side-effect that more irrelevant documents were retrieved at the same time. In other words, the new retrieval strategy was theoretically expected to perform an obvious lower precision value than the traditional strategy did. However, the evaluation results were in the contrary situation: the average precision of the new strategy was even slightly higher than the precision of the traditional strategy. It was because during the experiment, our system retrieved 0 relevant documents for some questions by using the traditional strategy. It means the traditional strategy performed 0 precision values while answering those questions. These situations then remarkably decreased the traditional strategy's average precision value. However, even if the new retrieval strategy leaded our system retrieved more not only the relevant documents but also the irrelevant ones; it still made sure there were at least some relevant documents retrieved so that the most precision values were bigger than 0. This was the reason our new document retrieval strategy performed a better recall value without losing the precision value comparing with the traditional strategy. Overall, our new document retrieval strategy helps our QA system retrieve more relevant documents to the question and same or less irrelevant documents than the traditional strategy dose.

As our QA system not only retrieves related documents but also ranks them for the users, it is not fair to evaluate it with the precision formula explained above. In the traditional precision formula, the number of retrieved relevant documents is divided by the total number of retrieved documents. From the uses' perspective, this precision describes the users' efforts which are spent on finding all the relevant documents among the entire retrieved documents. The users spend fewer efforts if the precision is higher. To the systems which do not rank the retrieved documents, the traditional precision formula is the proper way to evaluate the systems' precisions since the total number of entire retrieved documents is the number of documents the users have to go through to make sure they have viewed all the relevant documents. But for the systems like ours which presents the ranked retrieved documents in a descending order based on their similarities to the users' questions, users stop reading the retrieved documents after they have found the last relevant document. Thus, they only need to read parts of the retrieved documents in the ranked order to get the answers they need. Therefore, based on April and Scott's work [70], we have designed a new strategy to calculate the precision which also considers the ranks of the retrieved relevant documents from the users' perspective.

New Strategy of Calculating Precision

- IF : all the relevant documents in the collection have been retrieved by the system
- THEN:
    - Save the lowest rank of the relevant documents in the retrieved documents
    - Calculate the precision of the retrieval as:

      precision= $\frac{number\ of\ relevant\ documents\ retrieved}{the\ lowest\ rank\ of\ the\ relevant\ documents}$
- ELSE:
    - Calculate the precision of the retrieval as:

      precision= $\frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ documents\ retrieved}$

This new precision calculation strategy makes sure our system get a higher precision if and only if our system have successfully retrieved all the relevant documents it is supposed to retrieve. When it retrieves all the relevant documents, the number of retrieved documents the users have to go through is equal to the lowest rank of the relevant documents. If the system only retrieves parts of the relevant documents, the users have to read the entire retrieved documents to realize that some relevant documents are missing. Then we use the original precision formula to calculate the precision value of this case.

Based on the new strategy for calculating the precision, we have had a group of new precision values shown in Figure 11. Same with the two figures introduced above, the real line with diamond markers represents the new precisions of the new retrieval strategy which carefully expands the question; the dotted line with triangle markers denotes the precisions of the traditional strategy which searches the documents with the original query. The average of the new precisions of our new strategy was about 0.081. The average of the new precisions produced by the traditional retrieval strategy was around 0.07. There was an obvious increase from the original average precisions to the new calculated average precisions. It was because when our system had successfully retrieved all of relevant documents in the collection for some questions; it also ranked them with the higher ranks. The users then did not need to go through the entire retrieved documents to read all the relevant documents. They stopped checking the documents at the lowest rank of the relevant documents. Thus, the new precision should be higher since the number of retrieved documents the users read was less than the total number of retrieved documents.

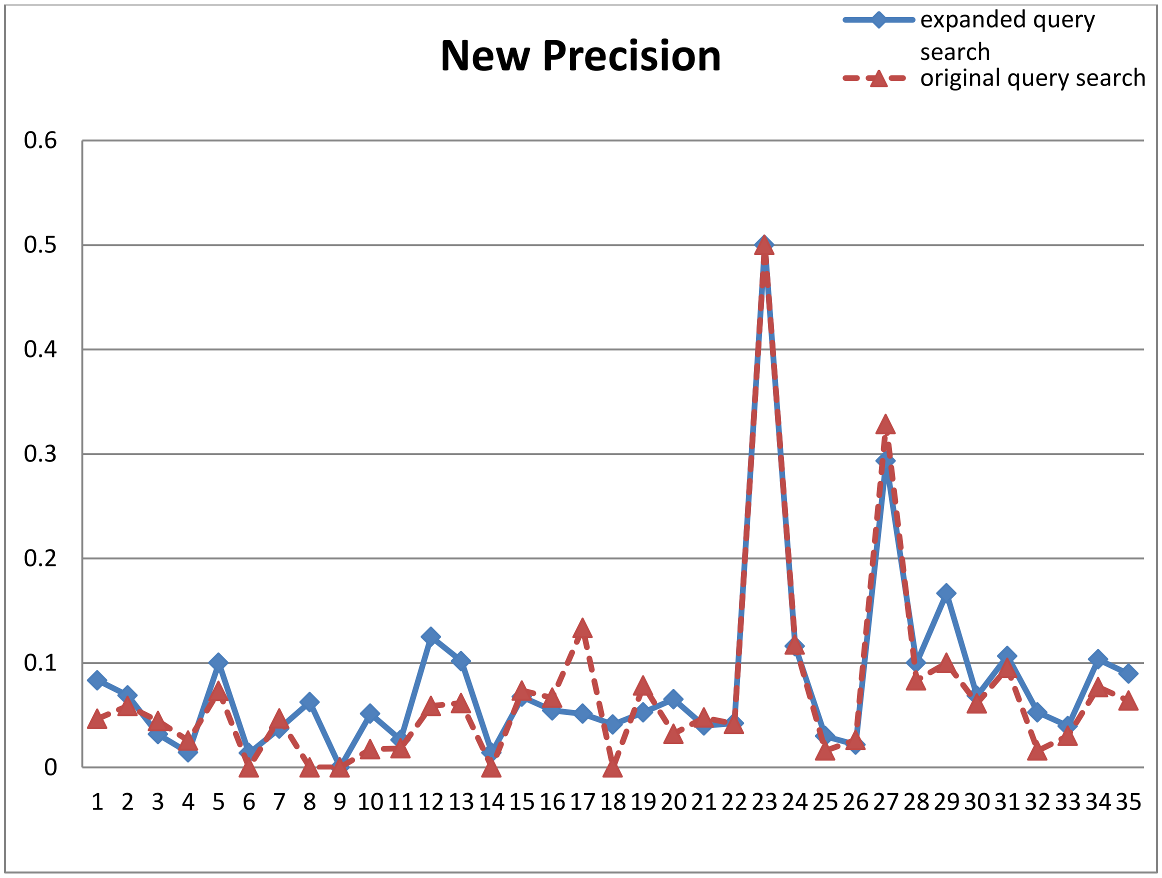


Figure 11

Based on the recalls and the new precisions we calculated while answering those 35 questions with the local document collection. Our new document retrieval strategy which expands the search queries with the snippets of their related web pages performed remarkably better retrieval results than the traditional document retrieval strategy did. The new strategy's expected lower precision as the trade-off of its higher recall did not appear since it made sure the system retrieve at least parts of the relevant documents back to the users. On the contrary, the traditional document strategy made the system to be over precise during the retrieval that none of relevant documents were searched in some cases. Furthermore, as both of the document retrieval strategies are the term-based search strategies which means as long as a document shares at least one common term with the question it will be retrieved, the higher recalls and the lower precisions are expected [70]. For the users who search a particular type of information such as the medical knowledge and the law cases in our system, the higher recall helps them find what they need eventually. The lower precision does not bother them much since they have to find the information they need no matter how long it takes.

Furthermore, we designed and conducted this evaluation to test the improvement of our new document retrieval strategy. During the ranking process, we did not use the MCCSM since the documents in the collection were not extracted from the related web pages. There were no corresponding web pages' ranks to use to estimate the qualities of the local documents. We could not use the documents' length as one of the factors to rank the retrieved documents since they did not use to be the paragraphs on some web pages. Thus, the evaluation results discussed above only describe the retrieval ability and improvement of our new document search strategy.

11.2 Evaluation on Overall Performance

Therefore, we have also designed another evaluation to test the whole performance of our QA system with the new retrieval and ranking strategies. There were two reasons we did not test the ranking algorithm MCCSM alone and compare the result with the CCSM's. First of all, it is because of the two modifications made in the MCCSM. One of them is adding their corresponding web pages' ranks into the similarity measurement to rank the retrieved documents. Another one is considering the length of each document during the ranking. As they were explained in the candidate answer ranking section, we treat those two new factors: web page rank and document length as two necessary but not the most important factors while calculating and ranking the similarity between each retrieved document and the question. The common terms and their weights shared by the retrieved documents with the question are still the key indicators determining the similarities. Therefore, in the new MCCSM formula, we carefully added those two new factors with the reductions of their powers to let them not affecting the similarity value dramatically. Because of these power reductions, the affections of those two new factors are harder to be detected separately during the evaluation to compare to the original CCSM's result.

Another reason we did not test the improvements of MCCSM separately in the next evaluation is the redundancy of the Internet [66]. As our system sends the users' questions to the Google search engine to acquire their related web pages as our knowledge resource, there is not only one correct answer extracted from the web pages. All the similar and correct answers are distributed into the different documents in our collection. Even though our new MCCSM produces the same candidate answers with the different ranks comparing with the traditional CCSM, it is possible that the MCCSM's correct answers' highest rank is equal to the highest rank produced by the CCSM. This possibility will lead us an inconspicuous evaluation result between the MCCSM and CCSM.

Therefore, we decided to evaluate the overall performance of our QA system instead. However, it does not mean we were denying the improvement of the MCCSM. On the contrary, we believe our QA system could produce the evaluation results below mostly because of the contribution of the MCCSM.

We submitted 50 questions which are written in natural English from the Text REtrieval Conference[2] (TREC) to our QA system in order to evaluate its overall performance. TREC offers numbers of different workshops to help researchers evaluate their information retrieval systems and compare the results with others'. There are professional scientists in the TREC who carefully design some different document collections and questions for the different research areas of information retrieval. Since we were going to evaluate our QA system's ability of improving the retrieval of information on the Internet, in other words, the ability of answering questions with the online information, we only used the questions and their standard answers from the TREC. We did not import the corresponding document collection into the system as the knowledge resource. To answer each question, our system dynamically built the document collection, retrieved and ranked the relevant documents from the collection. During the evaluation, we manually went through each group of the retrieved documents that our system produced. We searched the documents which contained the correct answer and recorded the highest rank those documents had for each question. The result is shown in Figure 12.

[2] http://trec.nist.gov/

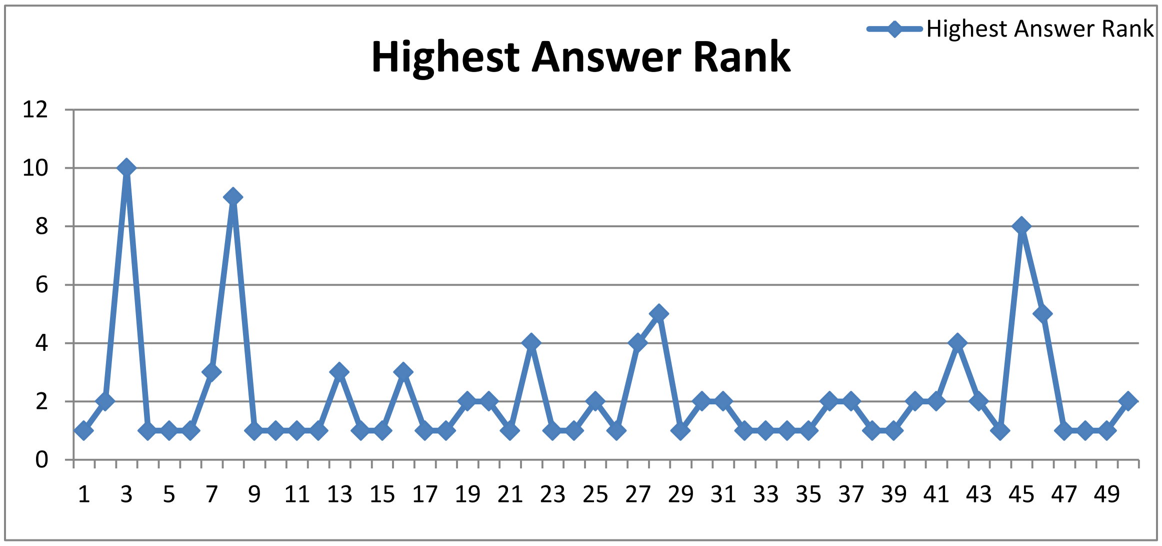


Figure 12

In Figure, the X axis represents the index of each question. The Y axis denotes each highest rank of the correct documents in the candidate answer to the each question. Apparently, most of the highest ranks were around 1 to 2. The average of the highest rank was 2.18. That means we usually found the first correct answer to the question in the top 2 to 3 documents after we submitted the question to our system.

Based on the result of this evaluation, it has been proved that the users who ask questions to our system will find the correct answers after they read the top 2 to 3 candidate answers provided by our system. In other words, our system is able to answer the users' questions in the top 2 to 3 candidate answers it retrieved back. As it was explained in the document preparation section, each document in our collection contains one text paragraph extracted from its related web page. It means the top 2 to 3 candidate answers the users need to go through to find their expected answers were the 2 to 3 paragraphs on the corresponding web pages. Comparing to the users who directly use the web page search engines to search their questions and go though several web pages to look for the answers, our QA system's users usually need to read only 2 to 3 paragraphs of texts to find the expected answers. Thus, our QA system has successfully improved the retrieval of the online information by reducing the users' searching efforts. This result also proves Jimmy and Boris' opinion [65] that to improve the search result of the web search engines, we can use some simple and affordable information retrieval techniques to produce a better search result. This evaluation result will help us set a threshold on the candidate answers for further reducing the users' reading and raising the quality of the users' experience eventually.

Finally, the reason we did not set an evaluation for our novel query refinement is that the improvements made by this step mostly depends on the users' knowledge. These are their choices to refine their queries for a better search results. An objective evaluation is not suitable for this subjective refinement.

12. Conclusion

In this paper, we have introduced and explained our QA system which was designed for improving the retrieval of information on the Internet. Several modified and improved algorithms were embedded into this system in order to attain this goal. In this section, we are going to summary the functions and the performances of the main components in our system and also have an overall conclusion of this complete system.

The initial motivation of our QA system was based on the web search engines' search process and results. The users submit their questions to those web search engines in order to search some useful answers. The search results they receive are mostly the lists of ranked links which lead them to some related web pages according to their questions. The users then have to click on those links and go through the corresponding web pages until they find the answers they need. In some worse cases, they even have to follow more links on the related web pages to track the answer. Based on the evaluation we made on the Google search engine, three of the top five web pages on the average directly answered our test questions. That means the users need to go into average two web pages to find the expected answers. Thus, the users are required to pay more efforts on the answer searching. And these web page link following and web page browsing reduce the quality of the users' answer searching experiences.

To improve these online information retrieval results, we have designed a QA system which directly represents the answers to the users based on the questions they ask. In order to deal with the users' questions written in natural English on the unlimited topics, our QA system dynamically connects to the Google search engine to access the abundant online information. Specifically, it receives a question from a user and sends it to the Google search engine as a piece of web search query each time. The Google search engine helps the system search for some related web pages based on the received search query and sends their links back. Based on those searched web links, our QA system visits each corresponding web page and extracts its text content. During the text extraction, the texts are downloaded based on their original paragraph structure on each web page. In other words, our system downloads all the text paragraphs on the related web pages to build up a temporary document collection as the knowledge resource for the received question. Furthermore, the system utilizes the Information Retrieval technology to retrieve parts of the documents in the collection which are relevant to the question and presents them back to the users as the candidate answers. During the document retrieval process, each document in the collection is determined by our system based on the common terms it shares with the question. If a document shares at least one common word with the question, it will be treated as a relevant document and sent back to the user as one of the candidate answers.

In order to efficiently locate and detect the terms contained by each documents in the collection during the common term matching process, our system indexes the entire document collection after they were extracted from the related web pages. During the indexing process, the system creates the term vectors to store the identical terms and the documents in which they appear individually. By the help of this index, the system does not have to go through entire collection each time to find the documents which contain the one or more common terms with the question. It simply checks the index terms at once, as long as there is a term matches with the term in the user's question, the documents' information of which this term occurs in will be detected and those documents then will be easily retrieved back.

During the term matching talked above, to avoid the type of situations which our system treats the terms "walk" and "walked" as two different terms, we have implemented the term stemming strategy into the document indexing process. Each detected term in the document will be first stemmed based on a list of English stems. For example, the term "knowledge" will be stemmed as "know" and saved as the same stem with the other terms such as "knew", "knowing", "known", etc. The information of the documents which contain all those terms will be saved together under the stem "know". Same with the document term stemming, the users' questions are also processed by the stemming strategy before the common term matching. Thus, the term “known” in the use’s question can be matched with the documents which contain the terms “knew”, “knowing”, etc. Therefore, our system is able to retrieve more relevant documents to the question.

However, matching more documents is not the ultimate goal of our system, matching the proper documents to the questions is. Based on the document retrieval strategy our QA system used, as long as a document shares one common term with the question, it will definitely be retrieved back to the user as a member of the candidate answers. Even the documents which only share the high-frequency terms such as “in”, “of”, “at”, etc. with the questions will still be treated as the relevant documents during the retrieval. Therefore, to reduce the disturbances of those high-frequency terms, our system ignores them during the document indexing based on a list of English high-frequency terms. Thus, even if the users’ questions still include those terms, they will not be matched since there are not such terms existing in the document index. This step was designed to solve the sensitivity of the original document retrieval strategy so that our system is able to retrieve less irrelevant documents back to the users.

By the help of our term-based document retrieve strategy which has been integrated with the term stemming algorithm and the high-frequency term elimination, our QA system is able to narrow down the Google search results for the users based on their questions. Instead of visiting and viewing several web pages to search for the useful information, the users who use our system only need to read a small number of retrieved documents which used to be the paragraphs on the corresponding web pages. Thus, our QA system carefully selects parts of the text contents from the related web pages and presents them to the users to reduce their readings.

Moreover, we have also noticed that the semantic information was unacceptably missing during the document retrieval process. For example: our system could not retrieve the document “Artificial intelligence is the intelligence of machines and robots and the branch of computer science that aims to create it” to the question “what is AI” since there is no common term shared by those two documents. Therefore, we have designed and embedded a new search strategy based on Mehran and Timothy’s document retrieval strategy [63] into our system.

We first proved that Mehran and Timothy’s strategy is not suitable for the systems which deal with the dynamic document collections. Based on their theory, to detect the semantic connection between two documents which do not share the common terms together, a system needs to expanse those two documents first with more related terms. Thus, the possibility of sharing the common terms will be remarkably increased from the value of 0 since more terms are added into those two documents. The two expansions of those two documents are made by sending them to the Google search engine separately and searching for their related web pages as the expansion materials. Because of the Google search engine offers the searched web page snippets which contain several segments matched to the search query, Mehran and Timothy use those snippets to expanse the documents. Thus, those two documents will be added with some high related terms coming from their related web page snippets. After the expansions, those two documents may share a few common terms and be matched during the document retrieval.

Therefore, Mehran and Timothy’s theory solves the semantic information missing situation of the traditional term-based document retrieval strategy with the simple but powerful expansions. However, it requires the systems expanse each documents in their collections for matching with the future questions. They have to send each document in their collection to the Google search engine to acquire the web page snippets as the expansion materials. For the systems which deal with the static document collections, they can follow this scenario and expand their documents at once before the answer retrieval. Then they only need to expanse the coming questions with the Google search engine each time before they are retrieving the related documents. But for the systems which are implemented with the dynamic document collections like ours, this search strategy is not affordable. Since a dynamic document collection constantly updates its document members, the new imported documents have to be expanded before matching with the new search queries. In our case, the system builds a completely new document collection based on each new question it receives. The average number of documents saved in our collection each time is around 350. It means our system has to send 350 documents to the Google search engine as the search queries to expanse each of them for the new question. Apparently, it is too expensive to accept. We also have proved the expansion of our document collection may lead the result that more irrelevant documents are retrieved since there are implicit connections between the question and our original documents in the collection.

Based on these two proves, we decided to only expand the users' questions each time since our system sends them to the Google search engine and receives their related web page snippets based on our original strategy. Based on the new expanded questions, the system then is able to retrieve the more relevant documents in the collection. Based on our evaluations, by using this new modified and improved document retrieval strategy, our QA system is able to produce a 1.5 times better candidate answers than the traditional strategy is.

We have also successfully implemented a new candidate answer ranking strategy in our system. In most cases, the users who submitted their questions to our QA system will receive a group of candidate answers back. They have to review each of the candidate answers until they find their expected information. Those candidate answers were retrieved by the system based on the common terms they share with the corresponding questions. Some of them contain much more related information with the question, others contain less. Our system ranks them based on the similarity between each of them and the received question. The ranked candidate answers will be presented to the users in the descending order of their similarity. The users then are able to view the answers with the better qualities first and find the correct answer earlier than going through the entire unranked candidate answers.

To implement this document similarity calculation, we first utilized the original CCSM formula which involves two main variables: the Term Frequency and the Inverse Document Frequency. Those two factors describe each term in the document index as how important it is not only in a specific document but also in the whole collection. We defined the described importance of a term as the weight of this term. Based on the combination of those two factors, the same term can have different weights in different documents since it occurs different times in each of them. An important term described by both of the factors should appear more in a particular document but less in the entire collection. Therefore, a retrieved potential document is evaluated by our system based on the common terms it contains with the question and the weights of those terms. A higher similarity value a document gets means this document is more related to the question. It then should be ranked higher in the candidate answer list to help the user find the expected answer as soon as possible.

Furthermore, based on the specialty of the documents in our collection, we have modified the original CCSM with two more variables. Since the documents were extracted from the web pages which were searched and ranked by the Google search engine based on the users' questions, the rank information then indicates the quality of both the web pages and the text contents they contain. The Google search engine gives a higher rank to a web page means this web page is more related to the search question. Thus, the text content in this web page is supposed to be more relevant to the question as well and has a bigger possibility to answer it. Therefore, our system saves the ranks of the searched web pages each document was downloaded from and used them into the similarity evaluation. A document, which came from a web page which got higher rank from the Google search engine based on the question, will be evaluated with a bigger similarity value. Moreover, our system predicts the qualities of the candidate answers in the way that a bigger document contains more relevant information. It is because the documents in our collection used to be the text paragraphs on the web pages. Based the feature of the online web pages, a longer paragraph has a bigger possibility to contain more important information to the search query. Therefore, the bigger documents will be attached with a higher similarity value and shown with a higher rank in the candidate answer list. Finally, since these two new variables are not the key factors affect the documents' similarities, we have reduced their affect abilities in the modified CCSM.

With the help of the similarity evaluations, the users receive a list of ranked candidate answers each time after they submit a question. They are expected to find the correct and useful information as early as possible. Our system then can be defined as reducing more readings of the users after narrowing the web pages reading into the paragraphs reading. Based on the evaluation of the system's overall performance, our QA system is able to answer the test questions with the top 2 to 3 retrieved documents on the average. That means only two to three paragraphs are needed to be viewed by the users who are looking for their answers. The conclusion has been easily made that our QA system has remarkably improved the retrieval of the information on the Internet.

To further improve the quality of the online information retrieval and the users' search experience, we have also expanded our QA system with a novel query refinement component. It offers the users an opportunity to refine their queries in three simple steps to receive the better search results. Comparing to the Google search engine's advanced search, there are a few improvements made. First of all, our system shows the users their previews search results while they are following the first step of the query refinement: reediting the question. From the previews search result, the users will have the deeper understandings about the powers of the terms in their original questions and make clearer and stronger questions this time. The users can also emphasize some of the terms in their questions which they think are more important during the retrieval by assigning numeric weights to them from 1 to 10. A term which is attached with a higher weight in the question will give the higher ranks to the documents which are related to it. The question term weighting refinement does not change the members of the candidate answers but their ranks. The last step of the query refinement is the Boolean query binding. The users can create a Boolean “NOT” query and bind it to the question for restricting the search result. The terms which are added in the Boolean “NOT” query will lead the result that the documents which contain some of those terms will not be shown in the ranked candidate answers. It is a powerful binding query the users can use to further filter the search results.

Those three steps of the query refinement effectively affect the members and the ranks of the future candidate answers. However, they are not mandatory to the users to use. Different combinations of them lead the users to the different search results. Since the refinement is mainly based on the users’ subjective opinions, we did not design and accomplish an objective evaluation on it. But we still strongly believe the specialty and the affectivity of this query refinement.

Thus, comparing to the Google search result which is a group of ranked web page links, our QA system has successfully helped the users narrow down the answer result into fewer related documents which used to be the parts of the paragraphs on the corresponding web pages. The users do not have to follow the web page links and browse the web pages in order to find the expected answers. Instead, they only need to directly read some retrieved documents. Furthermore, by ranking the similarity of the retrieved documents with the users’ questions, our system has further reduced the users’ readings into a small number of paragraphs. The improvement of the Internet information retrieval made by our system is clearly remarkable.

During the system designing, we have also modified and improved two traditional algorithms in the information retrieval area based on our case. We utilized Mehran and

Timothy’s simple and powerful document retrieval strategy as the basic concept and designed our own strategy to detect the semantic information in the document collection. The modified and improved strategy helps our system retrieve more semantically relevant documents with an acceptable time cost. With the deep understanding of the documents in our collection, we have also imported two new factors into the traditional CCSM to better rank the retrieved relevant documents for the users to view. Finally, the designing of the advanced search in our QA system based on the users’ perspectives helps them build the more robust search queries to retrieve the better groups of candidate answers.

Our system is a complete and functioning QA system. But there are still some spaces for us to improve in the future. So far, our system has been able to locate the correct answers in some retrieved short documents. As one of the future works, those retrieved documents can be broke down into sentences and only potential sentences will be combined together to compose the only one formal answer the users receive. Theoretically, this approach will involve the document clustering technique which is able to detect the qualified documents in the database and carefully combine them into a more complete answer based on the search question. It will help to further reduce the users’ readings and raise the precision of the system eventually.

Since our system connects the Google search engine to receive and improve its search result. More search options such as the specific web site search can be offered to the users in the future. Our system can be designed to restrict the Google search engine to only search the information in some specified web pages based on the users’ choices. Thus, the relevant documents extracted from a small number of web pages instead of the whole internet will increase the relevance of the future retrieved documents to the users' questions.

REFERENCES

[1] Gaurav Batra, Mansi Goel. An improved answer retrieval system taping the linkage structure for noisy SMS queries. International Journal of Computer Applications, 2012.

[2] Ryuichiro Higashinaka, Hideki Isozaki. Corpus-based question answering for why-questions. International Joint Conference on Natural Language Processing, 2008.

[3] Richard C. Wang, Nico Schlaefer, WilliamW. Cohen, Eric Nyberg. Automatic set expansion for list Question Answering. EMNLP '08 Proceedings of the Conference on Empirical Methods in Natural Language Processing Pages 947-954, 2008.

[4] João Silva, Luísa Coheur, Ana Cristina Mendes, AndreasWichert. From symbolic to sub-symbolic information in question classification. Artificial Intelligence Review, Volume 35 Issue 2, Pages 137-154, February 2011.

[5] Cheng-Lung Sung, Cheng-Wei Lee, Hsu-Chun Yen, Wen-Lian Hsu. An alignment-based surface pattern for a Question Answering system. Integrated Computer-Aided Engineering - Selected papers from the IEEE Conference on Information Reuse and Integration (IRI), July 13-15, 2008, Volume 16 Issue 3, Pages 259-269, August 2009.

[6] Tianyong Hao, Dawei Hu1, Liu Wenyin and Qingtian Zeng. Semantic patterns for user-interactive question answering. Semantics, Knowledge and Grid, 2006. SKG '06. Second International Conference, 2006.

[7] Leila Kosseim , Jamileh Yousefi. Improving the performance of Question Answering with semantically equivalent answer patterns. Journal: Data & Knowledge Engineering, Volume 66 Issue 1, Pages 53-67, July, 2008.

[8] Saeedeh Momtazi, Dietrich Klakow. A word clustering approach for language model-based sentence retrieval in Question Answering systems. CIKM '09 Proceedings of the 18th ACM Conference on Information and Knowledge Management, Pages 1911-1914, 2009.

[9] S. Kalaivani, K. Duraiswamy. Methodology for converting question to query form in Question Answering for automatic learning system. European Journal of Scientific Research, 2012.

[10] Raju Barskar, Gulfishan Firdose Ahmed, Nepal Barskar. An Approach for extracting exact answers to Question Answering (QA) system for english sentences. International Conference on Communication Technology and System Design, 2011.

[11] D.S. Wang. A domain-specific Question Answering system based on ontology and question templates. Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 11th ACIS International Conference, 2010.

[12] Shiyan Ou, Viktor Pekar, Constantin Orasan, Christian Spurk, Matteo Negri. Development and alignment of a domain-specific ontology for Question Answering. In Proceedings of the 6th Edition of the Language Resources and Evaluation Conference (LREC-08), 2008.

[13] Óscar Ferrández, Rubén Izquierdo, Sergio Ferrández, José Luis Vicedo. Addressing ontology-based Question Answering with collections of user queries. Lexical and Syntactic Knowledge For Information Retrieval, 2011.

[14] Hyo-Jung Oh, Sung Hyon Myaeng, and Myung-Gil Jang. Enhancing performance with a learnable strategy for Multiple Question Answering Modules. ETRI Journal, Volume 31, Page 419-428, August, 2009.

[15] Antonio Ferrández, Jesús Peral. The benefits of the interaction between data warehouses and Question Answering. EDBT '10 Proceedings of the 2010 EDBT/ICDT Workshops, Article No. 15, 2010.

[16] V. Rieser, O. Lemon. Does this list contain what you were searching for learning adaptive dialogue strategies for Interactive Question Answering. Natural Language Engineering archive, Volume 15 Issue 1, Pages 55-72, January 2009.

[17] Sebastian Varges, Fuliang Weng, Heather Pon-Barry. Interactive Question Answering and constraint relaxation in spoken dialogue system. Natural Language Engineering, Volume 15 Issue 1, Pages 9-30, January 2009.

[18] Wael Salloum. A Question Answering system based on Conceptual Graph Formalism. KAM '09 Proceedings of the 2009 Second International Symposium on Knowledge Acquisition and Modeling, Volume 03, Pages 383-386, 2009.

[19] Liang Zhenqiu. Design of automatic Question Answering system Base on CBR. 2012 International Workshop on Information and Electronics Engineering, 2012.

[20] Ulli Waltinger, Alexa Breuing, IpkeWachsmuth. Interfacing virtual agents with collaborative knowledge open domain Question Answering using wikipedia-based topic models. In proceeding of IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011.

[21] Detle Koll, Thomas Polzin. Providing computable guidance to releevant evidence in Question-Ansering system. Application Number: 13/025,051, Publication Number: US 2012/0041950 A1, Filing Date: Feb 10, 2011.

[22] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Philipp Cimiano. Template-based Question Answering over RDF Data. WWW '12 Proceedings of the 21st International Conference on World Wide Web, Pages 639-648, 2012.

[23] Elif Aktolga, James Allan, David A. Smith. Passage reranking for Question Answering using syntactic structures and answer types. ECIR'11 Proceedings of the 33rd European Conference on Advances in Information Retrieval, Pages 617-628, 2011.

[24] Arnaud Grappy, Brigitte Grau. Answer type validation in Question Answering Systems. RIAO '10 Adaptivity, Personalization and Fusion of Heterogeneous Information, Pages 9-15, 2010.

[25] Álvaro Rodrigo, Anselmo Peñas, Felisa Verdejo. Evaluating Question Answering validation as a classification problem. Language Resources And Evaluation, Volume 46, Issue 3, Page 493-501, 2012.

[26] Sofia J. Athenikosa, Hyoil Han. Biomedical Question Answering: a survey. Computer Methods and Programs in Biomedicine, Volume 99 Issue 1, Pages 1-24, 2010.

[27] Raju Barskar, Gulfishan Firdose Ahmed, Nepal Barskar. An approach for extracting exact answers to Question Answering (QA) system for English sentences. International Conference on Communication Technology and System Design, 2011.

[28] Ling Xia, Zhi Teng, Fuji Ren. Question classification for Chinese cuisine Question Answering system. IEEJ Transactions on Electrical and Electronic Engineering, Volume 4, Issue 6, pages 689–695, November 2009.

[29] Tiansi Dong, Ulrich Furbach, Ingo Glöckner, Björn Pelzer. A natural language Question Answering system as a participant in human Q&A portals. In proceeding of: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011.

[30] Sven Hartrumpf, Ingo Glöckner, Johannes Leveling. Efficient Question Answering with question decomposition and multiple answer streams. CLEF'08 Proceedings of the 9th Cross-language Evaluation Forum Conference on Evaluating Systems for Multilingual and Multimodal Information Access, Pages 421-428, 2009.

[31] Omar Trigui, Lamia Hadrich Belguith, Paolo Rosso. DefArabic QA arabic definition Question Answering system. In Proceeding of Workshop on LR & HLT for Semitic Languages, 2011.

[32] Ines Čeh, Milan Ojsteršek. Developing a Question Answering system for the Slovene language. WSEAS Transactions on Information Science and Applications, Volume 6 Issue 9, Pages 1533-1543, September 2009.

[33] Xavier Tannier, Véronique Moriceau. FIDJIWeb Question-Answering at Quaero 2009. European Language Resources Association, 2010.

[34] Jasmina Armenska, Aleksandar Tomovski, Katerina Zdravkova, Jovan Pehcevski. Information Retrieval using a Macedonian test collection of Question Answering. Second International Conference, ICT Innovations 2010, September 12-15, 2010.

[35] Dan Tufiş, Dan Ştefãnescu, Radu Ion, Alexandru Ceauşu. RACAI's Question Answering system at QA@CLEF2007. Cross-Language Evaluation Forum - CLEF, Page 284-291, 2007.

[36] Bob Coyne, Owen Rambow. LexPar: a freely available English paraphrase lexicon automatically extracted from FrameNet. Semantic Computing, 2009.

[37] Gosse Bouma, Geert Kloosterman, Jori Mur, Gertjan van Noord, Lonneke van der Plas, Jörg Tiedemann. Question Answering with Joost at CLEF 2007. Advances in Multilingual and Multimodal Information Retrieval, Page 257-260, 2008.

[38] Gosse Bouma, Gertjan van Noord, and Robert Malouf. Alpino: wide-coverage computational analysis of Dutch. Computational Linguistics in the Netherlands - CLIN, 2001.

[39] Olatz Ansa, Xabier Arregi, Arantxa Otegi, and Ander Soraluze. Ihardetsi a Basque Question Answering system at QA @CLEF 2008. CLEF'08 Proceedings of the 9th Cross-language Evaluation Forum Conference on Evaluating Systems for Multilingual and Multimodal Information Access, Pages 369-376, 2009.

[40] Mitchell Bowden, Marian Olteanu, Pasin Suriyentrakorn, Thomas D'Silva, and Dan Moldovan. Multilingual Question Answering through intermediate translation LCC's PowerAnswer at QA @CLEF 2007. Advances in Multilingual and Multimodal Information Retrieval, Pages 273 – 283, 2008.

[41] Marian Olteanu, Chris Davis, Ionut Volosen and Dan Moldovan. Phramer - An open source statistical phrase-based translator. Workshop on Statistical Machine Translation, 2006.

[42] Sofia J. Athenikos, Hyoil Han, Ari D. Brooks. A framework of a logic-based Question-Answering system for the medical domain (LOQAS-Med). SAC '09 Proceedings of the 2009 ACM symposium on Applied Computing, Pages 847-851, 2009.

[43] YongGang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James. Cimino, John Ely, Hong Yu. AskHERMES An online Question Answering system for complex clinical questions. Journal of Biomedical Informatics, Volume 44 Issue 2, Pages 277-288, April, 2011.

[44] Chuan Liang. Improved intelligent answering system research and design. Advanced Technology in Teaching - Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009), Page 583-589, 2009.

[45] Brian L. Cairns, Rodney D. Nielsen, James J. Masanz, James H. Martin, Martha S. Palmer, Wayne H. Ward, Guergana K. Savova. The MiPACQ clinical Question Answering system. In Proceeding of AMIA Annu Symp, Page 171–180, 2011.

[46] Cui Tao, Harold R. Solbrig, Deepak K. Sharma, Wei-Qi Wei, Guergana K. Savova, and Christopher G. Chute. Time-Oriented Question Answering from clinical narratives using semantic-web techniques. ISWC'10 Proceedings of the 9th International Semantic Web Conference on the Semantic Web, Volume Part II, Pages 241-256, 2010.

[47] Taniya Mishra, Srinivas Bangalore. Qme! A speech-based Question-Answering system on mobile devices. HLT '10 Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Pages 55-63, 2010.

[48] Samir Tartir, Bobby McKnight, I. Budak Arpinar. SemanticQA web-based ontology-driven Question Answering. SAC '09 Proceedings of the 2009 ACM symposium on Applied Computing, Pages 1275-1276, 2009.

[49] Maria Vargas-Vera, Miltiadis D. Lytras. AQUA: a closed-domain Question Answering system. Information Systems Management, Volume 27 Issue 3, Pages 217-225, 2010.

[50] Mahsa A. Yarmohammadi, Mehrnoush Shamsfard, Mahshid . Yarmohammadi, Masoud Rouhizadeh. SBUQA Question Answering system. Advances in Computer Science and Engineering, Page316-323, 2008.

[51] Anton Leuski, David Traum. NPCEditor: a tool for building Question-Answering characters. In Proceeding of Irec Conference, 2010.

[52] Tianyong Hao, Wenyin Liu, Chunshen Zhu. Semantic pattern-based User Interactive Question Answering User Interface design and evaluation. ICIC'11 Proceedings of the 7th International Conference on Advanced Intelligent Computing Theories and Applications: with Aspects of Artificial Intelligence, Pages 363-370, 2012.

[53] Ali Ghobadi Tapeh, Maseud Rahgozar. A knowledge-based Question Answering system for B2C eCommerce. Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference, 2008.

[54] Yi-Hsun Lee, Cheng-Wei Lee, Cheng-Lung Sung, Mon-Tin Tzou, Chih-Chien Wang, Shih-Hung Liu, Cheng-Wei Shih, Pei-Yin Yang, Wen-Lian Hsu. Complex Question Answering with ASQA at NTCIR 7 ACLIA. Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, NII, Page70-76, 2008.

[55] Jibin Fu, Jinzhong Xu, Keliang Jia. Domain ontology based automatic Question Answering. Computer Engineering and Technology, 2009. ICCET '09. International Conference, 2009.

[56] Ulrich Furbach, Ingo Glöckner, Hermann Helbig, Björn Pelzer. LogAnswer - a deduction-based Question Answering system. 4th International Joint Conference, IJCAR, 2008.

[57] Tom Yeh, John J. Lee, Trevor Darrell. Photo-based Question Answering. MM '08 Proceedings of the 16th ACM International Conference on Multimedia, Pages 389-398, 2008.

[58] Christina Unger and Philipp Cimiano Pythia. Pythia: compositional meaning construction for ontology-based Question Answering on the semantic web. Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems, Page 153-160, 2011.

[59] Lorand Dali, Delia Rusu, Blaž Fortuna, Dunja Mladenić, Marko Grobelnik. Question Answering based on semantic graphs. 2009.

[60] Ann Taylor, Mitchell Marcus, Beatrice Santorini. The Penn Treebank: an overview. 2003.

[61] William Tunstall-Pedoe. True knowledge open-domain Question Answering using structured knowledge and inference. AI Magazine, Volume 31 Issue 3, Page 80, 2010.

[62] Dmitri Roussinov, Weiguo Fan, Jose Antonio Robles-Flores. Beyond keywords automated Question Answering on the web. Communications of the ACM - Enterprise Information Integration: and Other Tools for Merging Data, Volume 51 Issue 9, Pages 60-65, 2008.

[63] Mehran Sahami, Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. WWW '06 Proceedings of the 15th International Conference on World Wide Web, Pages 377-386, 2006.

[64] Valentin Jijkoun, Maarten de Rijke. Retrieving answers from frequently asked questions pages on the web. CIKM '05 Proceedings of the 14th ACM International Conference on Information and Knowledge Management, Pages 76-83, 2005.

[65] Jimmy Lin, Boris Katz. Aranea: mining answers from the World Wide Web.

[66] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, Andrew Ng. Data-intensive question answering. In Proceedings of the 2001 Text REtrieval Conference (TREC 2001), 2001.

[67] C.J van Rijsbergen. Information retrieval (2nd edition). 1979.

[68] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. An introduction to information retrieval. 2008.

[69] Erik Hatcher, Otis Gospodnetic. Lucene in action (2nd edition). 2010.

[70] April Kontostathis, Scott Kulp. The effect of normalization when recall really matters. In proceeding of: Proceedings of the 2008 International Conference on Information and Knowledge Engineering, IKE 2008, July 14-17, 2008.

APPENDICES

Appendix A

A Report on Google Search Engine

Google as the most popular web searching engine is helping millions of users everyday with their various questions and interests. This report represents how Google search engine works based on three main tasks it accomplishes of. Unfortunately, core algorithms Google is utilizing with still understandingly are business secrets. Therefore, this report only refers to the general ideas and principles those algorithms are created based on.

1. Web Crawling

For the huge number of web pages on the Internet, Google needs to download every single one in term of covering all information users may need of. The procedural of downloading pages in a certain order in case of missing some of them is called web crawling. And the software that does this job is called web crawler. Since most of the web pages connect with others, this web crawler can be imagined as a spider walking on the spider web while collecting the information in the same time.

As matter of fact, Googlebot is the web crawler robot Google is using. It comprised of lots of computers to request and fetch different web pages simultaneously. This is how Google detects and saves documents through millions of web pages efficiently. Furthermore, Googlebot also considers about the visiting capacity of a web server. It usually sends requests to a server more slowly than its limit speed in case of the server is overwhelmed and dead for that.

As mentioned above, web crawler works like a spider though each page. Those web pages can be thought as different nodes which are highly connected. Therefore, mapping an efficient route through all web pages is crucial to a web crawler. Googlebot crawls with URLs and internal links. URLs usually indicate front pages and internal links help crawler find subpages. Googlebot is not a text based web crawler. Thus, there is no depth search or width search strategy. Instead, it more considers about how to keep data fresh (up to date).

Updating database is another biggest challenge Google deals with. The engineer in Google found that completely updating the database once a while does not work with the high speed of information growth. Therefore, they only update popular websites like news website constantly based on how often they get changed; there are more documents which works permanently are only saved once in the database of Google without updating again in a near future. They call this strategy as a fresh crawling.

2. Document Indexing

Google indexes downloaded documents from web crawler. This indexing process is seen as inversing the format of a document. Each word of documents is marked with the documents names. Therefore, a word may have different document numbers indicating different documents it belongs to. For example, the name: Bill Gates are indexed separately in different documents.

When Google receives a query asking about Bill Gates, it go though two indexing catalogues: Bill and Gates and compares the documents number in those two catalogues. Only documents with the number appearing both in Bill and Gates stack will be returned to users.

There are also certain words Google does not index with. Words which do not have a “real” meaning like: what, the, in, with, or are defined as stop words. Those stop words are not indexed by Google because they do not help with future searching.

3. Keyword Searching

Yes, Google still uses keyword searching. And Google has its own trademarked page ranker to rank the relevant pages for users. Unfortunately, no deeper details of core algorithms Google uses to deal with questions with nature language format and mapping them into documents are searchable through the internet. They are understandingly high business secrets of Google. However, it is known that there are around 200 factors involved into Googles’s algorithms. Those facts showing below are considered by google:

- Popularity of the page: how many links it is linked by other pages and how many links it link to others.

- The position, size and distance of the searching terms in the page: more close those key words are set a document, this document is more likely being placed in the top of the list.

- How long the Web page has existed: a newer (more updated) web page is more desirable than elder web pages.

Google also applies machine-learning techniques to improve its performance automatically by learning relationships and associations within stored data.

Since Google indexes HTML code rather than the text on the page, users can restrict searches on the basis of where query words appear. This function is included in Google advanced search. Once users get their first searching results back, they will have more chances to narrow the results based on the facts like: language, region, file type, etc.

Appendix B

A Report on Searching “How-To” Questions with Google Searching Engine

BACKGROUND

Nowadays, web search engines have been developed into some quick, accurate and powerful tools for people to search their interested topics. Google search engine is probably the most popular search engine in the last 5 years. As the influence of Google, users are used to search their varied questions with it and eager to find the best answers in a shortest time.

EXPERIMENT PURPOSE

Acquiring a general idea about how accurate and direct the searching results which are given by Google search engine are and which facts affect the quality of answers.

PROCESS

In this experiment, we used 20 “how-to” questions and searched the answers of them on Google Search engine. We only check top 5 web pages showing on the list. The pages were classified into two types: useful answers and direct answers. Useful answers are web pages which give users the answers eventually. However, users may need to click and go through several more links from there. Direct answers are answers users can find directly from the first page they entered in. No more links and searching needed.

Also, as a profit aspect, Google pops up some websites in the top of the searching list as advertisements. Those websites may not be the best match of the queries users just typed in. They are shown in the top of the list only because those websites pay Google for this position for gaining more attentions.

We considerate this situation in our results since most of users may not distinguish those commercial websites and regard them as the best match. This certainly affects the efficiency they are getting the answers.

Figure 1 showing below is the experiment result:

Question	Searching Time	Number of Results	Ad Pages	Useful Answers	Direct Answers
How to tie a tie?	0.34 sec	262,000,000	0	5	3
How to write a cover letter?	0.26 sec	65,200,000	1	5	2
How to download YouTube Videos?	0.27 sec	3,850,000,000	0	5	5
How to ride a hose?	0.26 sec	43,800,000	2	3	3
How to kiss?	0.28 sec	672,000,000	0	5	3
How to apply for OSAP?	0.39 sec	138,000	0	4	3
How to value a business?	0.35 sec	955,000,000	3	3	3
How to use Siri?	0.25 sec	53,800,000	0	4	3
How to eat healthy?	0.37 sec	105,000,000	3	3	2
How to overcome shyness?	0.19 sec	518,000	2	3	3
How to open RAR files?	0.26 sec	11,200,000	1	5	3
How to quit smoking?	0.35 sec	42,100,000	3	5	1
How to lose weight?	0.42 sec	233,000,000	3	3	2
How to poach an egg?	0.27 sec	1,130,000	1	5	4
How to love?	0.27 sec	5,680,000,000	0	1	1
How to negotiate salary?	0.26 sec	12,300,000	0	5	5
How to draw?	0.29 sec	742,000,000	0	5	0
How to flirt?	0.24 sec	96,100,000	0	4	4
How to increase metabolism?	0.28 sec	39,800,000	0	5	5
How to memorise spellings?	0.35 sec	2,440,000	0	5	4
Average	0.29 sec	643,376,300	0.95	4.15	2.95

Figure 1

* A webpage gives a direct answer is considered as a useful webpage as well.

* Advertisements in the top of the searching results are counted as top 5 web pages in our experiments.

RESULTS

In the general case, we found 3 pages (60%) giving the direct answers in the top 5 pages showing in the searching results. 4 pages (80%) give useful information about the question users ask. Also, the number of advertisement webpage being popped up at the top of the results by Google is 1.

During the whole experiment, we found there were several facts affect the quality of answers users get.

1. If the subject users ask for is more “instruct-able”, they probably get more specific answers or steps of instructions. For example, question “how to tie a tie” gets more direct answers from Google than question “how to love” since love is not a skill to learn.

2. If the subject is more specific, there is a bigger possibility for users to get direct answers. For example, question “how to poach an egg” has more direct answers than question “how to draw”. We need to make more choices in the web pages about painting to narrow the results of which kinds of picture you want to draw.

3. If the subject is more popular recently, users will get more useful web pages than some unpopular questions. For example: question “how to write a cover letter” gets more useful web pages than question “how to value business”.

4. The advertisement web pages affect the quality of answers a lot. Most of users do not know they are just some advertisements and regard them as the best match for their answers. Thus, this kind of websites gets more visitations. But the truth is, most of them are only relevant with the quires and they cannot offer user a direct answer.

Additionally, even the web pages give direct answers. Most of them have a few distracting advertisements in the middle of the answers. It is not easy and efficient for users to follow the instructions. Most of users describe this situation as “annoying”.

VITA AUCTORIS

NAME:	Ruoxuan Zhao
PLACE OF BIRTH:	Wulumuqi, Xinjiang, China
YEAR OF BIRTH:	1988
EDUCATION:	Wulumuqi No.1 middle school, Wulumuqi, Xinjiang,China, 2003-2006
	Tianjin University of Finance and Economics, B.MIS, Tianjin, China, 2006-2010
	University of Windsor, M.Sc., Windsor, ON, 2010-2012